

A11103 399458

NIST  
PUBLICATIONS

NISTIR 4330

**GRAPHICS  
STANDARDS IN THE  
COMPUTER-AIDED  
ACQUISITION AND  
LOGISTIC SUPPORT  
(CALS) PROGRAM  
FISCAL YEAR 1989  
Volume 2:  
MIL-D-28003  
Revisions,  
CGM Registration**

**DANIEL R. BENIGNI, Editor**

**U.S. DEPARTMENT OF  
COMMERCE**

**National Institute of  
Standards and Technology  
National Computer  
Systems Laboratory  
Gaithersburg, MD 20899**

**U.S. DEPARTMENT OF COMMERCE  
Robert A. Mosbacher, Secretary**

**National Institute of Standards and  
Technology  
John W. Lyons, Director**

**NIST**

QC  
100  
.U56  
#4330  
1990  
C.2

NATIONAL INSTITUTE OF STANDARDS &  
TECHNOLOGY

Research Information Center  
Gaithersburg, MD 20899

## DATE DUE

[illegible]

**GRAPHICS  
STANDARDS IN THE  
COMPUTER-AIDED  
ACQUISITION AND  
LOGISTIC SUPPORT  
(CALS) PROGRAM  
FISCAL YEAR 1989  
Volume 2:  
MIL-D-28003  
Revisions,  
CGM Registration**

**DANIEL R. BENIGNI, Editor**

**U.S. DEPARTMENT OF  
COMMERCE  
National Institute of  
Standards and Technology  
National Computer  
Systems Laboratory  
Gaithersburg, MD 20899**

**MAY 1990**



**U.S. DEPARTMENT OF COMMERCE  
Robert A. Mosbacher, Secretary**

**National Institute of Standards and  
Technology  
John W. Lyons, Director**





## EXECUTIVE SUMMARY

The overall objective of the Department of Defense Computer-aided Acquisition and Logistic Support (CALS) Program is to integrate the design, manufacturing, and logistic functions through the efficient application of computer technology. NIST has been funded since Spring 1986 to recommend a suite of industry standards for system integration and digital data transfer, and to accelerate their implementation.

During FY86 NIST tasks for CALS in the area of graphics standards focused on identification of recommended standards to OSD which would be applicable to the DoD environment; comparison of graphics standards among themselves and with product data standards; and the status of ongoing graphics standards efforts as well as related validation efforts<sup>1</sup>. CALS' needs for graphics standards were assessed, and an architecture for their inclusion into specific CALS programs was created. Finally, a plan was recommended for accelerating the development, related validation efforts and implementation of graphics standards into the CALS program.

Building on the knowledge and experience gained during FY86, NIST tasks for CALS in the area of graphics standards in FY87 emphasized the particular graphics standard dealing with the transfer of pictorial data from one system to another, namely the Computer Graphics Metafile (CGM) standard (FIPS PUB 128)<sup>2</sup>. Graphics tasks included an assessment of raster-to-vector conversion technology, and where the CGM might fit into that process; efforts toward development of CGM validation routines; injection of CALS requirements into the Extended CGM (CGEM) standards work, as well as into the CGM Registration process. In addition, functional requirements and conceptual design documents were completed for a reference implementation for CGM; a design specification was created for an IGES-to-CGM translator; and a preliminary CALS Application Profile for CGM was formed from the application profile work of the MAP/TOP organization.

During FY88, NIST tasks for CALS in the graphics standards area were in large measure a continuation of those efforts begun the

---

<sup>1</sup>Kemmerer, S., Editor, "Final NBS Report for CALS, FY86," U.S. Department of Commerce, National Bureau of Standards, NBSIR 87-3566, May 1987.

<sup>2</sup>Kemmerer, S., Editor, "A Collection of Technical Studies Completed for the Computer-aided Acquisition and Logistic Support (CALS) Program, Fiscal Year 1987," U.S. Department of Commerce, National Bureau of Standards, NBSIR 88-3727, March 1988.

year before<sup>3</sup>. Final text was completed for the initial publication of a Military Specification which is the CALS Application Profile for CGM. NIST continued to participate in graphics standards work in support of CALS requirements in the areas of CGM conformance testing, the CGEM, and CGM Registration. In the particular area of CGM conformance testing, the needs for testing both to FIPS 128 and to the Application Profile for CGM were identified; existing commercial implementations of CGM were analyzed and compared functionally to both CGM and Application Profile requirements; and required CGM conformance testing tasks were described in detail, responsibilities in the testing process delineated, and the impact of CGM testing was assessed both for CALS and the commercial marketplace.

This collection of reports represents the continuing efforts of the Graphics Software Group of NIST/NCSL in FY89 in support of computer graphics standards for CALS, and in particular CGM<sup>4</sup>. It provides a progress report on continuing graphics standards efforts related to the Computer Graphics Metafile (CGM) standard, including the Extended CGM (CGEM), Graphics Registration, and the CGM Application Profile for CALS (or MIL-D-28003). In addition, the creation of a Test Requirements Document for MIL-D-28003 is detailed. This Test Requirements Document will provide the basis for developing conformance tests to determine compliance with MIL-D-28003.

This report is subdivided into four separate final CALS deliverables, entitled as follows:

1. Test Requirements Document for CALS CGM Conforming Basic Metafiles
2. Injection of CALS Requirements in the Extended CGM (CGEM) Standards Work
3. MIL-D-28003 Revision Recommendations
4. CGM Registration in Support of CALS Requirements

---

<sup>3</sup>Morgan, Roy S., Editor, "A Collection of Technical Studies Completed for the Computer-aided Acquisition and Logistic Support (CALS) Program, Fiscal Year 1988, U.S. Department of Commerce, National Institute of Standards and Technology, NISTIR 4315, 4316, and 4317.

<sup>4</sup>The publishing of this collection of reports does not imply that the CALS Office has endorsed the conclusions or recommendations presented.

An additional deliverable completed for CALS by the Graphics Software Group during FY89 detailed the impact that two other graphics standards (namely PHIGS, or the Programmers Hierarchical Interactive Graphics System, and PIK, or the Programming Imaging Kernel)<sup>5</sup> will have on the CALS environment. It was published under separate cover, and is available through the CALS Policy Office or through the National Technical Information Service.

---

<sup>5</sup>Kemmerer, Sharon J., and Skall, Mark W., "Graphics Application Programmer's Interface Standards and CALS," U.S. Department of Commerce, National Institute of Standards and Technology, NISTIR 89-4199, October 1989.



### CONTRIBUTORS

NIST would like to acknowledge the major technical contributors to the separate reports contained herein. They are:

Peter R. Bono Associates, Inc., in particular Dr. Peter R. Bono, for his work on (1) above;

Henderson Software Company, in particular Mr. Lofton Henderson, for his work on (2) and (3) above, and

GSC Associates Inc., in particular Dr. George S. Carson, for his work on (4) above.

The editor would also like to gratefully acknowledge the efforts of those who participated in the review process of the documents presented in this report, namely:

David K. Jefferson  
Sharon J. Kemmerer  
Roy S. Morgan  
Susan Quinn Sherrick  
Lynne Rosenthal  
Mark W. Skall  
Jacqueline Schneider







FINAL REPORT

CALS FY89 SOW TASK 4.1.2

MIL-D-28003 REVISION RECOMMENDATIONS



## TABLE OF CONTENTS

1.	Summary & Recommendations . . . . .	1
1.1	Overview . . . . .	1
2.	Reconciliation of the TOP and CALS Profiles . . . . .	2
3.	CALS Profile Revision . . . . .	3
3.1	Activities . . . . .	3
3.2	References . . . . .	4
3.3	Procedural Considerations . . . . .	5
3.4	Criteria for Inclusion . . . . .	6
3.5	Addendum 1: Status & Recommendations . . . . .	9
3.6	Registration: Status & Recommendations . . . . .	12
3.7	Other Recommendations . . . . .	17
3.7.1	Baseline Document . . . . .	17
3.7.2	METAFILE VERSION . . . . .	18
3.7.3	METAFILE DESCRIPTION substring . . . . .	18
3.7.4	Escape -302 . . . . .	19
3.7.5	An Additional Hatch Style . . . . .	19
3.7.6	Text Precision & Append Text . . . . .	19
3.7.7	Private Additions to Parameter Lists . . . . .	19
3.7.8	Use of Font Indices . . . . .	20
3.7.9	Color vs. Black-and-White . . . . .	20
3.7.10	Temporal Priority . . . . .	20
3.7.11	I/O and Record Formats . . . . .	21
3.7.12	Levels and Structuring . . . . .	21
4.	The Presentation Problem . . . . .	21
5.	Future Work . . . . .	22
6.	Glossary . . . . .	23
	Attachment 1: Liaison Documents to TOP . . . . .	27
	Attachment 2: Liaison Letter Statement from ITRC to CALS DoD . . . . .	35





## 1. Summary & Recommendations

### 1.1 Overview

This report describes activities during FY89, presents final recommendations, and recommends future work for updating the CALS Application Profile (AP) of CGM, MIL-D-28003 (CALS SOW Task 4.1.2).

This task has consisted of two subtasks:

- A. Reconciliation of the CALS and TOP APs of CGM.
- B. Recommendations for modifications and extensions of the CALS AP.

Subtask A has resulted in substantial agreement between CALS and MAP/TOP technical personnel and is at a point where administrative action is required by both sides to create and execute the mechanisms for consolidating the projects. Further attention on this subtask is needed in FY90 to bring about a consolidation of the separate industry and government efforts.

Subtask B has resulted in a substantial body of recommendations for amendments to MIL-D-28003. These recommendations are detailed in the body of this report. There is also a plan for the updates to MIL-D-28003 which addresses:

- o timing, including coordination with activities in graphics standards;
- o body of content to be included;
- o mechanics and procedures of producing the new MIL-D-28003.

Some of the content of the MIL-D-28003 revision is being sponsored in Graphical Registration. Close liaison with the proposers has been maintained.

Finally, the recommendations in this report are not complete because:

- o Some content is pending completion of actions in graphics standards committees - these will be relatively routine.
- o Other content has been specified at a conceptual level but requires detailed formulation - these are items which derived from NIST/NCSL requirements studies produced late in FY89.

- o The work of this fiscal year has identified a body of changes larger than anticipated and resources were therefore not sufficient to cover all needed work.

This subtask will require additional work to produce the final text for revised MIL-D-28003 in FY90.

Because of slower than expected progress of needed CALS extensions in both the formal standards (the CGM addenda) and graphical registration, the criteria for inclusion of some extensions into the MIL-D-28003 revision have been changed. The new criteria insure adequate US national review but do not require the formal completion of the standards or registration process.

The extensions and additions proposed in the recommendations of this report derive from and logically follow events in the graphics standards committees. In this report the issues of procedures and timing for amending MIL-D-28003 have been investigated, and NIST/NCSL concludes that the most effective procedure will be one where the MIL-D-28003 review and amendment process can start at a time that is linked to milestones in the graphics standards process.

## 2. Reconciliation of the TOP and CALS Profiles

One of the goals of the Profile work from the beginning was a reconciliation of the differences between the TOP and CALS profiles.

The TOP CGM Application Profile was the starting point for the CALS profile. The NIST/NCSL representative participated in the review and refinement of the TOP profile until its completion in 1988. The CALS profile then had more than 6 months of additional review and refinement. As a result there are a number of discrepancies between the two profiles. Some of these are due to extensions and additional specifications in the CALS profile, such as additional engineering line types and hatch styles.

It has always been a goal of this project to keep the TOP and CALS profiles identical where they overlap. The proliferation of similar but slightly different "dialects" of CGM would cause confusion in the industry and should be avoided. The TOP and CALS constituencies are considered to be sufficiently similar that they might use the same profile, or at least use subsets of a single profile.

To explore this possibility the NIST/NCSL representative met with the TOP CGM expert and editor of the TOP profile, Mr. Kern Hardman. This meeting took place at the National Computer Graphics Association exposition in Philadelphia in late April 1989. After some discussion it was agreed that it was desirable to fold the TOP and CALS profiles into a single specification, perhaps with a separate TOP document pointing to the single specification and defining a subset.

As an aid in evaluating whether to pursue this merger, and as a necessary first step in defining the TOP document, a study of the differences between the two profiles was produced. This was sent to Mr. Hardman in early August. The study and cover letter to Mr. Hardman are in Attachment 1 of this report.

There has been subsequent communication with Mr. Hardman. The consolidation of effort is agreeable in principle to both parties. The preliminary technical content of the first MIL-D-28003 revision, as detailed in this report, is generally agreeable to Mr. Hardman. There are a number of procedural issues which must be resolved. The Information Technology Requirements Council (ITRC) of MAP/TOP has stated its position that the efforts of industry and government should be merged, and detailed its concerns over procedure, in a letter to the Director of the CALS office, Dr. Michael McGrath. This letter is contained in Attachment 2 to this report. Further action at the technical experts level is not likely to be effective until administrative arrangements are agreed.

### 3. CALS Profile Revision

What follows is a discussion of the activities during FY89 for the task of defining the next revision of the CALS profile. Criteria used to guide the selection of additions and changes to the profile will be presented. Finally all of the recommended functionality will be listed, indicating its status, and indicating what further actions are required before draft text can be produced.

During FY89 some procedural and timing questions were opened and resolved, and they will be discussed as well.

#### 3.1 Activities

The following work has been done to prepare the recommendations for revision of MIL-D-28003.



1. Assessed the 31 functions of CGM Addendum 1 for importance to CALS graphical interchange and produced sorting by inclusion or exclusion.
2. Contacted X3H3 registration experts at NIST/NCSL to determine the status and expected timetable of all of the CALS registration proposals currently in progress; verified and discussed US positions on these proposals in the ISO bodies.
3. Reviewed past and current CALS requirements studies.
4. Contacted NIST/NCSL CALS liaison to ascertain and decide policies, procedures, requirements, and timetables for the amendment process for MIL-D-28003.
5. Contacted NIST/NCSL CALS liaison and members of the CALS Industry Standards Working Group to attempt to ascertain what implications, if any, a pending 1840A "presentation problem" has for MIL-D-28003.
6. Correlated all potential extension items and modification suggestions with requirements and feasible timetables to produce the set of recommendations in this report.
7. Prepared interim report with preliminary recommendations, and circulated this report for comment within the CALS, graphics standards, and MAP/TOP communities.
8. Prepared this final report.

### 3.2 References

In the remainder of this report a few key documents are referred to:

1. CGM Addendum 1: the first set of formal standard extensions of CGM; recently completed substantive ISO processing; final text should be available near the end of the year or early in 1990.
2. Registration Proposals: exist in various X3H3 documents; are contained in the final report titled FINAL REPORT, CALS FY89 SOW TASK 4.3.4, CALS REQUIREMENTS FOR CGM.

### 3.3 Procedural Considerations

The content of the first revision to MIL-D-28003 depends to some extent upon the procedure by which the amendment is made, and upon the anticipated frequency of amendments. Because the functionality of the amendment is taken from work already completed or work in progress within the graphics standards bodies, the amendment procedure must be coordinated and correlated with certain milestones in the graphics standards bodies. The ideal timing of the amendments should be:

- batch of functionality reaches sufficiently stable milestone in graphics standards and registration processes;
- CALS required functionality extracted and included in proposed amendment;
- DoD, services, and industry review commences.

The extensions ideally are driven by the progress of needed items through the graphics standards process.

At the start of FY89 it was not known whether this method of proceeding would be acceptable within the established review and amendment procedures, or fit the goals and timetable of the CALS program. In particular the following information was needed before defining what the scope of the amendments should be.

1. How is MIL-D-28003 amended? What are the procedures and milestones in the process, and who are the key organizations and individuals?
2. How long is the process anticipated to take from commencement to conclusion?
3. Does CALS have a firm schedule for when the first set of amendments should be processed?
4. Is there a regularly scheduled amendment cycle, for example every year or every two years?
5. If not does CALS then have any requirements, guidelines, or criteria for minimum or maximum time between amendments or extensions?
6. Does Version 2 of MIL-D-28003 supersede Version 1 at some point in time, or do Version 2 and Version 1 both exist as legitimate interchange indefinitely? If the former, is there a period of overlap where both are acceptable or does Version 1 become obsolete and Version 2 operational on some definite fixed date?



7. Is there a requirement for backwards compatibility of the profile, implying for example that there can be no deletions from Version 1?

These issues were presented and discussed with the NIST/NCSL CALS liaison. Further discussion within NIST/NCSL and with the Services have resulted in these conclusions:

1. The MIL-D-28003 revision schedule will be coordinated with certain key milestones in the graphics standards and registration process over the next fiscal year, FY90.
2. The magnitude of the changes to the profile will classify it as a revision, and the resulting profile will have a new designation of MIL-D-28003A.
3. The requirement for backwards compatibility is not clear. However the desirability of such is clear and changes which create an incompatibility between the current and revised profile will be considered carefully.
4. There should minimally be two years between substantive revisions to the profile.

### 3.4 Criteria for Inclusion

There is a large number of potential extensions that are now or within 6 months will be at a sufficiently stable point in the standards and registration pipeline. In selecting those changes to go into the first revision of MIL-D-28003 NIST/NCSL has attempted to balance a number of sometimes conflicting criteria and guidelines.

Firstly, the proposal must be fairly stable. Ideally everything in the profile would be a part of the CGM standard or a completed standard extension. The delays in that process, particularly CGM Addendum 3, may make this level of stability impractical. A timeliness requirement may force inclusion of features ahead of their formal inclusion in the standard. If this is the case, then NIST/NCSL thinks that industry could tolerate the extensions being included in a preliminary formulation that is architecturally and functionally similar to what is expected to appear in the formal CGM extensions.

The implications to implementations of the profile leading the formal standards are as follows. Implementations would have to make an investment in new basic technology now. If the profile were amended again in a couple years when a piece of

functionality had completed formal processing and had become part of the CGM standard, then an implementation would have to revise mostly the interface to the piece of basic technology. What this means is that the parameterization or precise structure of the particular functionality would change in the final version, and would require some modification to the metafile encoder or decoder of the implementation. This would be a minor effort relative to the initial effort to implement the basic technology, since the major part of the implementation effort is devising and including the basic technology.

While it would be preferable to have no changes in the future to extensions being currently included, NIST/NCSL feels for any given functional extension that one change of such a nature will be acceptable. NIST/NCSL does not think that two changes would be; it will create too much confusion in the industry. This could happen for example if a GDP (Generalized Drawing Primitive) were included as a private GDP in MIL-D-28003, and then a changed formulation completed graphical registration and were adopted in an amendment, and then yet another different formulation were included in the formal CGM extensions that comprise Addendum 3 and adopted in a second amendment.

With these principles in mind, almost all of the functionality which NIST/NCSL recommends for MIL-D-28003A is:

1. In the CGM standard or the first standard extension, CGM Addendum 1; or
2. Will be in CGM Addendum 3 in some form; or
3. Has completed Graphical Registration or has reached a "stable" milestone in Graphical Registration.

Concerning the second and third points, note that a number of CGM extensions identified in CALS requirements studies are being promoted through the standards and registration process as part NIST/NCSL's work on CALS.

The qualifier "almost all" needs explanation. Some features that were included in MIL-D-28003, in particular a few of the hatch styles, have been rejected by the Graphical Registration Authority in its first consideration (all of the rest of the line types and hatch styles were approved). A narrow view of what comprises a "hatch" was taken, and NIST/NCSL thinks the decision will be reversed. But in any case, the styles were identified in CALS requirements studies, and if the ISO committees refuse the proposals then they should stay in the profile as private types and values.



The definition of "stable" as it is used above needs definition. Ultimately no proposal is stable until it is approved by the relevant ISO body and is "on the books." The US position on registration is that the ISO Graphical Registration Authority should only reject or modify a proposal if it is flawed, incorrect, or unworkable. By this principle, once the US takes the proposal out of X3H3 (the domestic graphics standards committee) to ISO it should then be stable. This is the criterion NIST/NCSL will apply for inclusion of registration proposals in the amended MIL-D-28003. The US procedures generally result in two rounds of review in X3H3. This should take a year or less (but its timeliness is dependent upon continuity of efforts to "shepherd" the proposals once they are formulated).

In selecting this definition the alternatives were considered. There are two earlier milestones that could potentially be chosen as the inclusion milestone for CALS extensions:

- the initial formulation of the proposal itself;
- the end of the first round of review within X3H3.

NIST/NCSL thinks both of these are inadvisable and likely to introduce errors and incompatibilities into the profile. This is no reflection upon the talent of the proposer or the quality of that work. Rather the current set of graphics standards and related specifications is so broad and so interwoven and interrelated that review by experts and specialists in all of the disciplines of X3H3 is necessary in order to assure that the proposal is both correct and consistent with related work. NIST/NCSL therefore recommends the inclusion guideline:

**Recommendation:** a proposed functional extension is stable enough for inclusion in MIL-D-28003 if either the item is a registration proposal with two rounds of review in X3H3, or the item is in the final text of CGM or one of its addenda, or the item has completed the registration process in ISO.

While NIST/NCSL does not preclude the inclusion of something less stable, the requirement should be compelling if this is considered.

MIL-D-28003A will contain additions to MIL-D-28003 and may also delete some specifications or alter some specifications. NIST/NCSL believes that backward compatibility is a strong goal (if not a requirement, see previous section) and so deletions and changes are approached with some caution. There is no question

if there is a mistake or defect to be fixed. Other cases must be approached on a case-by-case basis. If a specification is not being used or implemented, then there is little problem with deleting or changing it. If it is in use then the justification for change should be strong.

One final criterion which deserves some discussion is timing of amendments and changes. There are two aspects to this issue: when should the first amendment begin review; and how often should major amendments be anticipated. NIST/NCSL recommends that:

**Recommendation:** the first amendment should commence review after final text of CGM Addendum 1 is produced, and after the second batch of registration proposals completes its second X3H3 review (see below).

**Recommendation:** a major upgrade about every two years is acceptable; more frequently would be too demanding on the industry.

### 3.5 Addendum 1: Status & Recommendations

The first formal standard extension to CGM, Addendum 1, completed its final ISO ballot in June. Preliminary final text has been produced and is being reviewed for accuracy for 6 weeks. Early indications are that a second 6-week review will be required, commencing sometime in November. Final text should be available by January 1990.

There are two basic pieces of functionality in CGM Addendum 1 that CALS has been interested in and has been promoting:

- o **Global Segments:** this is an internal symbol definition and instancing capability. This was identified as a high priority item in FY87 CALS requirements studies.
- o **Closed Figure:** this is a primitive that is a composite of other primitives and which is a filled primitive like POLYGON. Lines, arcs, rectangles, polygons, etc., can be strung together to form a complex shape which is treated as a single primitive.

Altogether there are 31 new functions in CGM Addendum 1. Several are required to implement each of the above two capabilities. Several others will be required in order to make the needed features work properly.

**Recommendation:** the features of CGM Addendum 1 should be adopted in MIL-D-28003A as indicated in Table 1. Estimated date of stability: January 1990.



TABLE 1. Recommendations - CGM Addendum 1 Function List

Addendum 1 Element	Recommendation	Notes
Begin Segment	include	1
End Segment	include	
Pick Identifier	exclude	2
Copy Segment	include	
Inheritance Filter	include	3
Clip Inheritance	exclude	
Segment Transformation	include	
Segment Highlighting	exclude	
Segment Display Priority	include	
Segment Pick Priority	exclude	
Segment Priority Extent	include	
Maximum VDC Extent	include	
Device Viewport	include	4
Device Viewport Specification Mode	include; limit to 0,1	5
Device Viewport Mapping	include	
Line Representation	exclude	6
Marker Representation	exclude	6
Text Representation	exclude	6
Fill Representation	exclude	6
Edge Representation	exclude	6
Line Clipping Mode	include; limit to SHAPE	7
Marker Clipping Mode	include; limit to SHAPE	7
Edge Clipping Mode	include; limit to SHAPE	7
Begin Figure	include	
End Figure	include	
New Region	include	
Connecting Edge	include	
Save Primitive Attributes	exclude	
Restore Primitive Attributes	exclude	
Circular Arc Center Reversed	include	

Notes:

- Both Global Segments and Local Segments should be allowed.



2. As CGM Addendum 1 states, the meaning of PICK IDENTIFIER is not standardized and is by agreement between generator and interpreter. Its use is application dependent and it should therefore not be included in the profile.
3. For the way that internal symbols are most likely to be used in engineering applications, both values of the INHERITANCE FILTER are needed. For value STATE\_LIST, the defined symbol has no attached attribute but inherits it from the context into which it is instanced. For example one might define a colorless symbol and then set the color just before instancing the symbol into a picture. For SEGMENT, the symbol has the attributes in effect at the time of its definition. For example a certain dash pattern might be required for some lines within the symbol. Inheritance is specified on a per-attribute basis, and it is easy to postulate examples that require mixing modes within one symbol (e.g., combine the two examples above).
4. This should supersede the Escape -302 of MIL-D-28003. It is precisely the same function. Since this Escape has never been seen in practice, NIST/NCSL thinks it is safe to simply remove it from the profile rather than leaving it in and deprecating its usage.
5. Value 0 is "fractional", like the current Escape -302. Value 1 is "millimeters". Value 2 is native device units, which is device dependent and should be disallowed in the profile.
6. While these functions provide something of a useful macro capability for some attributes, no applications in the US are using bundled attributes and no requirement for these capabilities has yet been identified by CALS.
7. There appears to be no need for LOCUS clipping (see Addendum 1 text) but there is definite need for SHAPE clipping. Unfortunately the Addendum 1 default is LOCUS so these elements must be included to override the default.

In addition to the new functions of CGM Addendum 1 there are some new rules for using existing functions:

1. COLOR TABLE and PATTERN TABLE can occur in the Picture Descriptor as well as the Picture Body, and use within the Picture Body is discouraged. This solves (in Addendum 1) the problem that MIL-D-28003 dealt with by restrictions on placement and usage of the two elements.

2. Several of the Picture Descriptor elements - COLOR SELECTION MODE, LINE WIDTH SPECIFICATION MODE, MARKER SIZE SPECIFICATION MODE, EDGE WIDTH SPECIFICATION MODE - may now appear in the Picture Body. This is required in order to make Global Segments work properly. With the exception of the color mode, which can be tedious to switch, the elements are not particularly difficult for implementations to deal with in a Picture Body.

### 3.6 Registration: Status & Recommendations

There are three batches of registration proposals currently in progress and a fourth being formulated. The content of each is looked at below and a decision proposed whether it should be included in MIL-D-28003A or deferred.

#### Batch 1.

These comprise the engineering line types and hatch styles which are already included in MIL-D-28003. All of these have now been approved by the ISO authority except for some of the hatch styles. Those hatch styles have been rejected because they did not meet the ISO definition of hatch.

The ISO authority should be assigning positive registration indices within a few months. MIL-D-28003A should use the positive indices. Because no instance of the negative indices of original MIL-D-28003 have been encountered, NIST/NCSL feels that the negative values can be safely prohibited rather than allowed but deprecated.

**Recommendation:** The negative indices in MIL-D-28003 be replaced with the positive ISO register indices where applicable. Expected timetable: May 1990.

"No." in the following table heading refers to the registration proposal number.

TABLE 2. Batch 1 Registration Proposals

No.	Description	Recommendation	Notes
2	Linetype, Single Arrow	change index	
3	Linetype, Single Dot	change index	
4	Linetype, Double Arrow	change index	
5	Linetype, Stitch Line	change index	
6	Linetype, Chain Line	change index	
7	Linetype, Center Line	change index	
8	Linetype, Hidden Line	change index	
9	Linetype, Phantom Line	change index	
10	Linetype, Break Line 1	change index	
11	Linetype, Break Line 2	change index	
12	Hatch Style, cast iron ...	change index	
13	Hatch Style, steel	change index	
14	Hatch Style, bronze, brass, ...	change index	
15	Hatch Style, white metal, zinc, ...	change index	
16	Hatch Style, magnesium, aluminum, ...	change index	
17	Hatch Style, rubber, plastic,...	change index	
18	Hatch Style, cork, felt, ...	change index	
19	Hatch Style, sound insulation	undetermined	1
20	Hatch Style, thermal insulation	change index	
21	Hatch Style, titanium ...	change index	
22	Hatch Style, concrete	undetermined	1
23	Hatch Style, marble, slate, ...	change index	
25	Hatch Style, rock	change index	
26	Hatch Style, sand	change index	
27	Hatch Style, water and other liquids	undetermined	2
28	Hatch Style, with grain wood	undetermined	2
29	Hatch Style, across grain wood	undetermined	2

## Notes:

1. Was rejected by ISO; continue to use the negative index of MIL-D-28003.
2. Was rejected within X3H3 and never forwarded to ISO; continue to use the negative index of MIL-D-28003.



## Batch 2.

This batch of proposals covers

1. finer drawing control (line cap, join, etc.) such as found in various proprietary systems and Page Description Languages (PDLs);
2. user-defined linetype;
3. advanced geometry (splines, conics, etc.).

The proposals have been through two X3H3 ballot and review rounds. The resulting proposals are now being sent to ISO. The ISO ballot is expected to end sometime in February 1990. It is expected that these can be included in this revision of MIL-D-28003. All of these are high-priority CALS items, and all are in draft Addendum 3, in substantially the same form (except for user line type).

**Recommendation:** The registration proposals in "batch 2" should be included in MIL-D-28003A as indicated in Table 3. Estimated date of stability: November 1989.

"No." in the following table heading refers to the registration proposal number.

TABLE 3. Batch 2 Registration Proposals

No.	Description	Recommendation	Notes
1	Linetype for Setdash	include	1
36	Set Dash	include	1
37	Set Line Cap	include	
38	Set Line Mitre Limit	include	
39	Set Line Join	include	
40	Cubic Bezier	include	
41	Conic Arc	include	
42	Set Conic Arc Transformation Matrix	include	
43	Parametric Spline Curve	include	
44	Rational B-Spline Curve	include	

### Notes:

1. The two functions comprising the user-defined line type are formulated quite differently from the draft Addendum 3 work. NIST/NCSL thinks that the problems arising from an



architectural distinction in Addendum 3, which will be stable within 2 years, justify using the Addendum 3 formulation, with a negative ESCAPE id, rather than using the registration proposal in this case.

### Batch 3.

This batch contains proposals for improvement of CGM text capabilities. These are high priority changes. Getting high quality text in metafile pictures is currently one of the most difficult things to do and one of the most important for publication quality graphics.

Unfortunately this batch is also the furthest from completion in the X3H3 review process. There has been one ballot and review round. There were heavy comments, and these will likely result in significant changes to the proposals. The comments have been returned to the proposer. If the proposer is able to modify the proposals by the time of the Nashua X3H3 meeting in September, then the second X3H3 review round could take place before the late January 1990 X3H3 meeting, and it is possible that there would be a set of final proposals to go to ISO around February.

According to the 2-review criterion noted above, it would then be February 1990 before any proposals were mature enough to put into the revisions of MIL-D-28003.

**Recommendation:** improvements to text and font capability are sufficiently important that production of draft text for MIL-D-28003A should be delayed until appropriate registration items are advanced enough and FY90 work supplies other needed definitions. Estimated timetable: February 1990.

There was strong liaison between the Registration Item proposer, the NIST/NCSL representative, and the X3H3 CGM experts group before and at the September X3H3 meeting in Nashua, NH. NIST/NCSL feels that enough consensus was achieved that the text capabilities of MIL-D-28003 can be improved as per the above recommendation. (Note: At the Nashua meeting there was extensive work on the US position on the ISO Working Draft of Addendum 3. The text needs of CALS as reflected in the registration proposals should be nearly identical to what the US wants in Addendum 3.)

NIST/NCSL thinks it likely that a phased approach to improving the MIL-D-28003 text capabilities is most likely to succeed. Under a phased approach it should be possible to identify a relatively small number of not too difficult extensions that would greatly improve the capabilities for many applications.

These would go into MIL-D-28003A. Then a thorough and complete set of capabilities would go into the next revision, MIL-D-28003B. The time frame for MIL-D-28003B is likely to be roughly when work on Addendum 3 is substantially complete.

NIST/NCSL understands that it is likely that the proposals in the third batch will be substantially revised by the proposer. Therefore NIST/NCSL is recommending "include when stable" with the understanding that at least some of the high priority text items should go into this revision of the profile.

"No." in the following table heading refers to the registration proposal number.

TABLE 4. Batch 3 Registration Proposals

No.	Description	Recommendation	Notes
45	Edge Mitre Limit	include when stable	
46	Edge Cap	include when stable	
47	Edge Join	include when stable	
48	Set Italic Text	include when stable	
49	Set Outline Text	include when stable	
50	Set Shadow Text	include when stable	
51	Set Underline Text	include when stable	
52	Set Bold Text	include when stable	
53	Set Fully Justified Text	include when stable	
54	Set Condensed Text	include when stable	
55	Set Extended Text	include when stable	
56	Pel Array	include when stable	1
57	Set Indexed Color Response	exclude	2
58	Set Direct Color Response	exclude	2

Notes:

1. This is substantially the same as the current CGM CELL ARRAY except that compression techniques (CCITT Group 3 and Group 4, and LZW) are added. CGM binary encoded CELL ARRAY already has some compression capabilities. These are being used effectively in real world applications - NIST/NCSL has seen raster images of 512x512, both index and direct color, encoded in 30K-50K bytes. Given this fact, the Pel Array may not be as high priority as some other items. This will need to be coordinated closely with the X3H3 CGM experts, who are working on Addendum 3. This latter work is including tiling capabilities such as those found in the ISO 8613/8 tiling addendum and in MIL-R-28002. The topic of color compression needs some work as well.



2. The need for advanced color capabilities in MIL-D-28003 is uncertain. It appears that the majority of use is going to be for bi-level black & white or grayscale work. Accordingly NIST/NCSL thinks these proposals can be assigned lower priority.

In addition to the attribute selection capabilities of these proposals the revised text functionality of MIL-D-28003A should:

1. allow in metafiles and mandate for interpreters the fonts Helvetica, Times Roman, Courier or their metric equivalents. These are copyrighted or trademarked names but virtually every publishing system has the metric equivalent available.
2. define how to use the FONT LIST to name a font resource at varying levels of specificity, from the capability to unambiguously name a particular resource to the capability to call out a class of fonts based on some attributes (e.g., state that any "serif bold" font will do).
3. define how to use the attributes of the above registration proposals, within the body of the metafile, in combination with the FONT LIST mechanism.
4. define how to use symbol and character sets that do not fit well within the ISO 2022 model of character sets and character set switching.
5. define for metafiles and interpreters how to specify character codes for the dozen (approximately) character sets needed to support IGES drawings and SGML fonts.

These topics were discussed at the Nashua meetings. Except for the last two, detailed recommendations can now be prepared based on existing material in registration, draft Addendum 3, and NIST requirements studies.

### 3.7 Other Recommendations

#### 3.7.1 Baseline Document

The original CGM standard was published in both a US version and an ISO version. The US version is an ANSI standard ANS X3.122-1986. The ISO version is designated ISO 8632/1-4:1987. They are identical except for document style and layout. The ANSI document was adopted by FIPS PUB 128, which in turn is the baseline document that is referenced in MIL-D-28003. X3H3 is

currently working on getting domestic CGM work redesignated as an "international project". By these procedures the main technical work takes place in ISO committees with US participation. US public review of the resulting milestone ISO documents is automatic, as is the adoption of the final result as an ANSI standard.

In reality this is exactly how work has been taking place for the last several years. Giving the project this new designation will mean that X3H3 does not have to have a US document editor to revise the ISO document stylistically, and does not have to conduct tedious parallel voting and review procedures. In the end there will only be one document, an ISO document.

X3H3 intends X3.122-1986 to be replaced by ISO 8632/1-4:1987 and CGM Addendum 1 to be handled by the new procedures. NIST/NCSL will be revising the FIPS designation of CGM to reflect this, but will not do so until X3H3 has obtained formal approval. In the interim:

**Recommendation:** MIL-D-28003 should be revised to point to the appropriate ISO documents for CGM and addenda until they get FIPS designations.

### 3.7.2 METAFILE VERSION

The MIL-D-28003A will include some of the functionality of CGM Addendum 1. Metafiles conforming to CGM Addendum 1, and using new capabilities, will have a METAFILE VERSION value of 2. The original CGM is a proper subset of CGM Addendum 1 (proper terminology is uncertain here - CGM Addendum 1, or CGM-plus-Addendum-1, or...). A legal ISO 8632:1987 metafile will be a legal ISO 8632/AD.1:1989 metafile. Such a file may have METAFILE VERSION value of 1.

**Recommendation:** The Basic Values of METAFILE VERSION shall be 1 or 2.

### 3.7.3 METAFILE DESCRIPTION substring

Metafiles corresponding to MIL-D-28003 must contain a substring "MIL-D-28003/BASIC-1" (case insensitive) in the string of METAFILE DESCRIPTION, to identify the metafile as a CALS metafile. This requirement will need to be revised.

**Recommendation:** The METAFILE DESCRIPTION element shall contain the substring "MIL-D-28003A/BASIC-1".



#### 3.7.4 Escape -302

The current Escape -302 is a device viewport, identical to a function which will be adopted from CGM Addendum 1.

**Recommendation:** Remove Escape -302 from MIL-D-28003.

#### 3.7.5 An Additional Hatch Style

Registration proposal #59 defined an additional hatch style which consists of a pattern of alternating dots. The proposal has been accepted by X3H3.

**Recommendation:** Include the hatch style of registration proposal #59 in the Basic Set of MIL-D-28003A.

#### 3.7.6 Text Precision & Append Text

The CGM standard allows TEXT PRECISION to be changed between partial text strings which are appended. The intended results are nearly impossible to figure out, and will be highly implementation dependent. MIL-D-28003 should prohibit this.

**Recommendation:** The TEXT PRECISION element shall not occur between non-final TEXT, RESTRICTED TEXT, or APPEND TEXT and following APPEND TEXT in conforming basic metafiles.

#### 3.7.7 Private Additions to Parameter Lists

Files have been encountered in which private parameters have been added to parameter lists of elements. This occurs on elements with fixed-length parameter lists, and the Parameter List Length of the binary element header is adjusted appropriately to include the private data. The CGM standard does not specifically prohibit the practice, but there are ample indications that it should not be legal. The Metafile Maintenance Rapporteur Group is likely to rule that this is illegal. In the meantime:

**Recommendation:** The Parameter List Length of elements in conforming basic metafiles shall be as indicated in the tables of the CGM Binary Encoding (Part 3); in particular no private parameters shall be added to elements.

### 3.7.8 Use of Font Indices

One of the biggest impediments to predictable interchange is the failure to define required font indices.

**Recommendation:** All font indices which are explicitly used in the metafile shall be defined in the FONT LIST.

### 3.7.9 Color vs. Black-and-White

The necessity of supporting a rich color model has been questioned. Much technical publishing and engineering drawing work needs no more than bi-level black & white, or multi-level grayscale. It is uneconomic to force all CALS-compliant suppliers to have full color capability.

**Recommendation:** MIL-D-28003A should have mechanisms and structure so that full color, black & white, and grayscale implementations are all recognized as conforming implementations.

**Note:** this is one aspect of a more general "levelling" or "structuring" issue which is being raised now about CALS conformance.

### 3.7.10 Temporal Priority

Many metafiles, particularly those coming from graphics arts packages, assume that the recipient device is a "temporal priority" device. This means that objects occurring later in the file overlay and obscure objects occurring earlier in the file. This assumption works fine on raster display monitors, raster driven slide cameras, and various high quality plotters. But on certain COM devices which are not driven from a frame buffer and on pen plotters such as those sometimes used in drafting and design this assumption causes problems. A descriptor element has been proposed which would declare whether the file contents assume temporal priority or not. There has not yet been a concrete proposal for either registration or addendum 3.

**Recommendation:** MIL-D-28003B should have an ESCAPE, which occurs in the Metafile Descriptor, which declares whether temporal priority is assumed for successful interpretation and/or rendering of the file.

### 3.7.11 I/O and Record Formats

MIL-D-28003 inherited from TOP certain specifications concerning the size of records and tape blocks for metafiles. This specification has been widely criticized. It does not appear to belong in MIL-D-28003 itself, but is more properly dealt with at the next higher level - MIL-STD-1840A - or left to supporting data transport services in the computing support environment.

**Recommendation:** The record formatting specifications of MIL-D-28003 should be removed, and the specifications of MIL-STD-1840A ensured to be adequate for successful interchange over the specified media.

### 3.7.12 Levels and Structuring

It is now becoming clear that it will not be economic for the CALS program to force conformance of all suppliers to all possible specifications in the second version of the profile, MIL-D-28003A. There are two particular areas where this is true: color vs monochrome; and character set, glyph collection, font support. By example, a system which is designed to handle the equivalent of IGES engineering drawings should be required to carry the full color capability and dozen SGML glyph collections that a high quality technical illustration system might have to support. To force conformance of all suppliers to all possible specifications will unnecessarily force up the government's acquisition costs for the components to implement CALS systems. However, any leveling or structuring must be designed clearly and minimally so that it does not require technical expertise for procurement officers to specify in contracts what is needed in a system.

**Recommendation:** MIL-D-28003A should have some minimal packaging, levelling, or structuring to account for the different application requirements that are all being addressed by CGM.

This aspect of the revision will be addressed in FY90.

## 4. The Presentation Problem

A potential interchange problem was raised in a series of communications early in FY89. These communications were between members of the CALS Industry Standards Working Group. The problem can be summarized as a "presentation problem." Briefly the issue is: how are the separate contents, such as CGM illustrations, IGES drawings, etc., assembled and positioned within viewports in the complete document?



During this fiscal year, the NIST/NCSL representative began to contact people in the working group who are interested in the problem and are addressing it. Results are incomplete at this time. Progress on this question will be entirely dependent upon the pace of an ad hoc group which has been set up to investigate it, and which has not yet met.

**Recommendation:** The work of this ad hoc group should be tracked in FY90 and any implications for the profile should be accounted for in MIL-D-28003A.

## 5. Future Work

The recommendations of this report should be implemented in FY90 by:

1. by following the appropriate work in the standards and registration bodies;
2. producing draft text for MIL-D-28003A;

In addition, the CALS/TOP reconciliation and consolidation should continue to be pursued at the technical level once agreement is established at the administrative level.



## 6. Glossary

AFNOR	The French equivalent of ASC X3H3.
ANS	American National Standard, the final stage in the ANSI pipeline, nothing remains but possibly the printing.
ASC X3H3	Accredited Standards Committee X3H3, the ANSI accredited committee responsible for computer graphics standards in the US.
BSI	British Standards Institute, the British equivalent of ASC X3H3.
CGEM	Computer Graphics Extended Metafile, a set of addenda and extensions to CGM, being processed by ISO, currently nearing DP stage.
CGI	Computer Graphics Interface, another ANSI/ISO standards project, currently at the 2nd DP stage. CGI is an interface standard which exists about at the level of the CGM in the graphics reference model (device level). CGI is an interactive (input) and highly extended and enriched interface specification, whereas CGM has output-only functionality (for picture definition) and is a picture description protocol (a graphical database). CGI embeds CGM output functionality as a subset.
CGM	Computer Graphics Metafile, ANSI standard X3.122-1986 and ISO standard ISO 8632/1-4 1987.
DAD	Draft Addendum, the same as DIS, but for an addendum as opposed to a standalone project.
DIN	The German equivalent of ASC X3H3.

DIS	Draft International Standard, the project stage in the ISO pipeline after DP. The technical content of the project is supposedly highly stable and it is expected that IS text can be produced subsequent to processing the DIS ballot results.
DP	Draft Proposal, the second stage in the ISO processing pipeline. After national bodies have commented on the WD, it is altered and refined and then registered as a DP. Another round of ballot and comment takes place on the DP.
GDP	Generalized Drawing Primitive, a CGM element which is a catch-all for geometric primitives which are not standardized in the CGM standard itself. These are private or user defined primitives. GDPs also may be submitted for graphical registration.
GKS	Graphical Kernel System, an application programmer interface to computer graphics, now an ANSI and ISO standard.
GKSM	A metafile for use with GKS. One was proposed in non-standard Annex E of GKS. Work on it was deferred in favor of CGM, and now of extended CGM (CGEM).
Graphical Registration	A process by which private geometric primitives (GDPs such as Pel Array), private control functions (ESCAPes such as Set Dash), and private types (e.g., a private line type) is given a fixed identifier and entered in a central ISO register. The item thereby becomes accessible in a uniform manner to all users, and can be thought of as having "semi-standard" status.
IS	International Standard, the final stage in the ISO pipeline, nothing remains but possibly the printing.
ISO TC97/SC21/WG2	The predecessor to SC24 (prior to December 1987).

ISO/IEC JTC1/SC24	International Standards Organization, Joint Technical Committee 1, Standing Committee 24, the international counterpart to X3H3.
MRG	Metafile Rapporteur Group, the sub-group of WG3 responsible for CGM maintenance and CGM extensions.
PDAD	Proposed Draft Addendum, the same as DP, but for an addendum as opposed to a stand alone project.
PHIGS	Programmers Hierarchical Interactive Graphics System, an application programmer interface to computer graphics, with 3D, structure hierarchy, etc., meant to be highly dynamic. It is nearly completed as an ANSI and ISO standard.
WD	Working Draft, the first complete draft of a proposed ISO standard, the starting document for subsequent work and review.
WG3	The working group of SC24 responsible for standards work in metafiles and device-level interfaces, i.e., CGM and CGI.
X3H3.3	The subcommittee of X3H3 that is responsible for CGM and CGI.





**Attachment 1:**  
**Liaison Documents to TOP**



4 August 1989

Mr. Kern Hardman  
Boeing Computer Services  
P.O. Box 24346  
Seattle, WA 98124

Dear Kern,

I'm writing in my capacity as a CALS consultant on maintenance and extension of MIL-D-28003, the CALS application profile of CGM. I'd like to follow up on our conversation at NCGA, when we discussed how to maintain consistency between the MAP/TOP and CALS application profiles. To summarize that conversation: we concluded that maintenance of separate and complete documents for the TOP CGM AP and the CALS CGM AP (MIL-D-28003) no longer appears desirable.

This is not to say that the APs should be identical. That should only be the case if our respective user communities are identical and have identical requirements. We have not concluded that to be the case. However, our respective user communities are very similar, and we did conclude that where their functional requirements overlapped, then the AP specifications pertaining to those requirements should be identical. In other words, one AP might contain capabilities or specifications that the other does not contain and does not intend to contain, but where the APs overlap they should be identical.

We have really had this goal from the beginning, as the CALS community reviewed and participated in the specification of the TOP profile, and vice-versa. We have not quite realized our goal. This is probably inevitable given our distinct processing methods, review procedures and timetables. In these circumstances it is inherently difficult to maintain two complete documents and keep them identical in overlapping areas.

There are a two practical possibilities for maintaining a single document. The first: we could try to encapsulate the APs of the TOP and CALS communities as distinct specifications in the single document. The second: we could write the document to specify the "larger" AP and have the second AP be a "delta document" against the complete document.

From a procedural point of view, the second approach appears most likely to achieve our goals. The principle reason is that it allows the two communities to continue to have their autonomous review

cycles and procedures, and at the same time it ensures that the profiles stay highly compatible because there is ever only one complete document.

At this point the CALS AP appears to be the larger AP, i.e., it has more specifications and restrictions, and has added more functionality as well. From our conversations, it appears that likely that this will continue to be true. We anticipate that the CALS AP will be ammended in the next year or so to add specifications from CGM Addendum 1, from Graphical Registration, etc.

If we agree to this approach, MIL-D-28003 would become the base document for both the CALS AP and the TOP AP. The current TOP AP would be replaced by a document that points to MIL-D-28003, and specifies changes, deletions, and limitations. The TOP community would participate in the revision of the base document by forwarding new requirements to CALS, and proposed ammendments to the CALS AP would be circulated to the TOP community for review and comment. When MIL-D-28003 is ammended then the TOP community would decide how and when to revise its profile to be based on the new common document. This should work smoothly, particularly as we both have a strong goal of keeping the revised profiles backwardly compatible with previous versions.

The question that we must now address is: how do we implement this? I think there are two steps: firstly, we reaffirm that we intend to proceed in this way; secondly, the TOP AP would be replaced with a new document which points to the MIL-D-28003 document.

I think having an exact list of differences between the current version of the two APs would help in both steps: it will let you evaluate the implications of making the change; and it will give you the basic "raw material" for writing the delta document. Accordingly, I have reviewed and compared both documents and produced a list of differences of substance. The list is enclosed. For each list item, I have included a brief assessment of implications for TOP -- whether the item seems like a good idea in general or is really CALS-specific.

Could you please, at your earliest convenience, get in touch and give me your current thoughts on the matter, and your ideas on how best to proceed? I think this is something that we should be able to take care of fairly easily, with beneficial results for our two closely related user communities.

Sincerely yours,

Lofton R. Henderson  
President, Henderson Software



Differences between the TOP and CALS  
Application Profiles

1. The Metafile Description element in TOP contains the substring TOP/BASIC-1, whereas in CALS it contains the substring MIL-D-28003/BASIC-1. Assessment: both TOP and CALS should keep their unique substrings to identify the profile.
2. The Metafile Description element in CALS is required to have a substring identifying the source (company and product). In TOP this is only a suggestion. Assessment: experience has shown id information to be very valuable. It should be valuable for TOP.
3. TOP allows interpreters to use only the Hershey fonts for rendering pictures. CALS allows the use of any font which is "metrically identical" to a requested Hershey font. Assessment: the CALS specification is formulated to give identical results, metrically, to the TOP, and it does give implementors more freedom and the option of producing better quality. Its adoption should be seriously considered for TOP.
4. CALS has added a number of additional line types and hatch styles for engineering applications. Assessment: CALS identified a requirement for these. TOP has not identified such a requirement, and so might well eliminate these for the TOP AP.
5. CALS has specified two conformance levels for interpreters (Publication Level and Draft Level), whereas TOP specified only one. Assessment: The two levels arose in CALS as a result of considering the implications of forcing all interpreters to implement the profile fully, for complete uniformity and predictability. This would essentially mean: no fallbacks, no monochrome interpreters, etc. For publication this is in fact required. For development it seemed excessive, and would make all interpreters more expensive and resource intensive. CALS thinks that the distinction is a useful one. TOP should consider having this specification.
6. CALS and TOP differ somewhat on their reference to Annex D. CALS goes into more detail, and makes distinctions based on Draft Level versus Publication Level. TOP doesn't define conformance level precisely (does it intend something more like Publication or Draft? It appears the former), and which fallbacks of Annex D are allowed is not completely clear. Assessment: The intent appears to be the same. The effective specification perhaps would be identical if the APs both had the same conformance level structure. If TOP adopts the two-level structure, the references could be identical. If not, then the TOP AP would

need its own Annex D reference, which presumably would be a clarification of the current one.

7. There is a note in CALS suggesting that the first three elements of the metafile be, in order, Metafile Version, Metafile Element List, and Metafile Description. This apparently will not be a feature of CGM Addendum 1 as originally anticipated, but is still thought to be useful. Assessment: it is useful, and in its current form of a non-binding suggestion should not cause any problems for TOP.
8. In CALS the behavior of the Device Viewport escape has been clarified and brought into alignment with the specifications of ISO CGI and the ISO CGM Addendum 1. It is not clear whether this is slightly different from what is intended in TOP or simply a more precise statement of the same requirement. Assessment: we assume the latter, and so this should not be objectionable to TOP.
9. The maximum string length is 256 in CALS and 254 in TOP. 256 is somewhat of an arbitrary choice, being based around some perceived limitations of popular computer systems. However some systems actually cannot go longer than 255. The longest string expressable in a short-format CGM binary text string is 254. Assessment: there does not seem to be any requirement for 256 over 254; compatibility between TOP and CALS should pertain in this case.
10. The Basic values of Transparency in CALS have been limited to the single value 1 (on). The effects of this element in CGM itself are described as device dependent for the value 0 (off). Assessment: no requirements for value 0 (off) have been stated, and a lot of experience has failed to reveal a single usage of the feature in practice. Value 0 (off) should be eliminated from the Basic Set.
11. CALS has dropped the requirement that the Metafile Descriptor contain the Character Set List element listing (0,4/1) and (1,4/2). The vast majority of applications will simply use ASCII, and that is the default, so announcing the requirement for the other set is misleading. Assessment: it is thought that this TOP specification was intended to effectively require that interpreters support both sets. This is already achieved by the table in 6.2.6.2. Since it presents misleading information to the recipient of the metafile, TOP should drop the specification.
12. TOP says all Color Table elements must appear before the first graphical primitives. CALS now says that they just must appear before they are first used by a primitive or attribute.



Assessment: the effect (preventing color dynamics) is the same but the CALS specification is easier for many applications and generators to implement; TOP should have this specification.

13. The previous distinction applies to Pattern Table elements as well. Assessment: same.
14. There is no requirement for color index definition in the metafile in TOP. CALS now requires that either all indexes which are used in a metafile be defined, or none be defined. The case of having some indexes which are used be defined and some not defined is prohibited. Assessment: experience in the NCGA Integrate demos has shown that partial definition of color indices is a real contributor to unpredictable and sometimes inexplicable results. For the goal of interoperability TOP should have this specification.
15. The TOP AP defines the default Color Table for interpreters to be a cyclic repetition of the colors Red, Green, Blue, Yellow, Magenta, Cyan, Black, White, starting at index 2. Indexes 0 and 1 are left undefined. In the default case CALS now specifies the same table except that index 0 is defined to be white and index 1 black. CALS allows these two to be swapped in Draft Level interpreters. Assessment: the reasons for this specification are the same as the reasons for the previous specification. For predictable interchange TOP should have this specification.
16. CALS has added an Escape element, Implicit Color Table, with Escape Indicator -303. This element allows a generator to specify how interpreters are to initialize their color tables. One option is "none", which means the interpreter is to use its native color capabilities as best it can and initialize its color table accordingly. The other two options are shorthands for selecting one of two useful initialization schemes -- the first is the "cyclic" scheme (the default for the AP), and the second is a uniform sampling of the RGB color cube. Assessment: this was thought to be a useful feature in CALS; TOP should assess whether it is useful for its constituency.
17. CALS has defined how metafile color requests should be mapped by Draft Level interpreters when the output device is only capable of two levels (foreground and background). Assessment: this removes an ambiguity. Some implementations have made bad choices here. The specification improves interoperability, so should be useful to TOP.
18. CALS specifies what shall be the clipping limits in all cases. Assessment: this removes a CGM ambiguity and improves interoperability, so should be useful to TOP.



19. CALS specifies that line types and edge types shall be continued across interior vertices of a polyline or polygon. Assessment: this removes a CGM ambiguity and improves interoperability, so should be useful to TOP.
20. In CALS, at Publication Level all text must be rendered at stroke precision. In TOP, all metafiles must have an item in the Metafile Defaults Replacement which sets text precision to stroke. Assessment: it is thought that the CALS specification achieves what was intended by the TOP specification (but which is not achieved, because the default can be circumvented by an explicit precision setting in the picture body).
21. The TOP document has some description of generation and interpretation of metafile from a system architecture standpoint. Assessment: there is no problem with this remaining in TOP if CALS is used as the base document for TOP.
22. CALS has a specification that the character set selected shall be representable in the font selected. Assessment: this is not achieved in either CALS or TOP now, because Hershey cannot represent complete ASCII; it is nonetheless a desirable goal and should be stated.
23. CALS has a section correcting known errors in the CGM standard. Assessment: this is useful to all CGM users.
24. TOP and CALS both say that Metafile Defaults Replacement shall not be partitioned. TOP further says that no element within MDR shall be partitioned. Assessment: CALS should say this as well. It will be put into the next revision. TOP should keep the specification.
25. TOP has a section on metafile transfer using FTAM and MHS. Assessment: this was not appropriate for the current CALS environment. There is no problem with TOP retaining the specification.

**Attachment 2:**

**Liaison Letter Statement from ITRC to CALS DoD**





**-DRAFT-**

**"USER COLLABORATION: A Proposal to Improve the Efficiency of Information Technology Standards Selection for Government and Industry"**

Prepared for Dr. M. McGrath, Director, CALS Office  
by the MAP/TOP Division of The Information Technology Requirements Council

**EXECUTIVE SUMMARY**

The voluntary standards process in the United States has been changing over the last several years. One of the more noticeable changes has been the involvement of the user community in what has been a vendor dominated activity. This user push is based on user requirements to apply information technology to achieve competitive advantage in a world market. The government as a user shares the same need as industry. Both groups need to be able to easily share useful information electronically within and between organizations on a global scale. Both groups need solutions that are cost effective, that can evolve to meet changing needs and that will preserve the existing investment made in computing equipment and applications software. This cannot be done without the acceleration of strategic standards. There is a genuine concern today that the United States is falling behind Europe and Japan in developing and promoting the use of international standards.

Today, there exists two or more separate processes for this user push. One process focuses on industry users, the others on the government's needs. This paper proposes a collaboration of government and industry to pursue the mutual objectives of the information technology "user".

## BACKGROUND

Since the early '80's, the United States voluntary standards process for information technology standards has been changing. A number of issues provided the catalyst for these changes.

The development of base standards was vendor dominated. Users and user requirements were noticeably absent in the process.

The government, both civilian and military agencies were being pushed by congress to use the voluntary industry standards instead of developing their own.

The development cycle for standards and for conforming products took way too long.

Products developed to these base standards frequently did not work together. (x.25 is a good example).

The changes that began to occur as a result of these issues are for the most part, very positive.

NBS, now NIST initiated a series of workshops to provide an open forum for vendor implementation agreements. These agreements - the refining of a stable base standard - were a necessary step if compatible products were to be built.

The MAP/TOP User Group was formed. The primary objective of this multi-industry user group is to accelerate the availability of information technology standards based products that meet real user needs. Activities include a consensus based process to define core sets of user requirements, carry forward those requirements into appropriate activities, including the NIST agreements process, and document the final results in specifications that can be used to procure systems that meet the established requirements.

*comment: Various terms have been used to define this user process: profiling, tailoring, and flavoring are all terms used to mean the reduction of a general base standard into a more implementable specification.*

The Corporation for Open Systems (COS) was established to develop the tests necessary for testing and certifying products that are compliant to the standard specifications.

The Government OSI User Group was formed under the auspices of the Department of Commerce. This group generally performs a similar set of functions as MAP/TOP for the government. The specification produced becomes a Federal Information Processing Standard that is used as a procurement specification. The current version GOSIP, version 1, FIPS 146 is a compatible subset of the MAP/TOP specification work.

The DOD CALS Program was initiated as a major program by the Department of Defense. This activity is broader in scope than MAP/TOP and GOSIP because it covers areas beyond the one of accelerating standards conforming products. CALS is intended to accelerate the use of electronic information exchange between military agencies and their suppliers and contractors. One aspect of the CALS



program does focus on the documentation of MILSPEC standards for data interchange, again a tailoring of a base standard to meet the military agency user requirements. In addition, CALS will use FIPS 146 to define the network products and application services to exchange information in the formats specified by the CALS interchange standards.

### **Parallel Activities**

The MAP/TOP activity, the GOSIP activity, and parts of the CALS activity all have common elements

All three are user driven activities.

All focus on the use of information technology standards produced by the voluntary standards process.

All groups define user requirements, or assume a knowledge of user requirements, tailor or profile base standards to meet those requirements, and document the results in specifications used to procure systems.

All groups are attempting to accelerate the availability of compliant products.

### **Current Benefits**

The results of all of these activities, NIST, MAP/TOP, COS, GOSIP, and CALS, are positive. The output of all three user group activities, are reasonably consistent. User input is made into the standards process and the agreements process. Output of the agreements process is the basis for the user specifications, and a trial test activity is underway with a good start on the tests necessary to certify products as MAP, TOP and GOSIP compliant.

## **THE PROBLEM**

From a user perspective, the current parallel user activities present some problems.

- 1 ) Because there are three separate groups, each establishing requirements and publishing specifications, few vendors or users not involved in all the processes understand that the work is complementary and compatible - at least today. The result of this confusion is product delay. Vendors don't know which specifications to use and users don't know what to buy.
- 2 ) There is no established process to keep future work aligned.
- 3 ) The process is inefficient. Multiple committees are addressing the same issues, schedules are different and extra steps are inserted in the process to harmonize the results and make them technically consistent.
- 4 ) The multiple committee process forces skills dilution affecting the quality of the output.
- 5 ) While the MAP/TOP committees are open to everyone and all work is distributed for comment, the government specification development is done in closed committee with only the final work published for comment. This difference in process makes 'harmonizing' difficult.



- 6 ) Worst of all, as users we are not leveraging and pushing the market the way we would as a combined force. As a unified effort, products would be available faster and at lower cost to everybody.

## **PROPOSED SOLUTION**

The proposed solution consists of three areas of collaboration: planning, specification development, and joint promotion.

### **Planning**

Each group has one or more activities planning new work items in the information technology standards area. This planning appears to be done on a yearly basis. To improve the process, a recommended logical first step is to form a joint planning activity which would meet at least annually.

Each group would come prepared with its work items

Work would be split into common activities and organization unique activities.

The schedules for common activities would be reconciled.

Shared committees would be formed to address common activities identified.

Each organization would pursue their unique activities independently.

### **Specification Development**

The joint or shared committees would:

Compile requirements to meet needs of government and industry.

Identify any special requirements that would prevent development of cost effective products - *hopefully these are a few isolated cases.*

Reach agreement on how special requirements are to be handled.

Carry forward consolidated requirements into other appropriate forums.

Develop common text to be used in all specifications.

Distribute agreed upon common text to be used in all specifications for public comment and review. This can be done in one of two ways:

- 1 ) Each group conducts independent comment and review process.
- 2 ) A combined comment and review process is used.

Modify text based on valid comments.

Each group would then proceed to publish specifications using their normal procedures and processes, using common text and cross-referencing each other.

## **Promotion**

In order to insure that maximum benefit and impact is achieved, joint promotion should be planned. Activities could include:

- joint press releases
- shared sessions at conferences
- cross-reference other groups activities whenever possible
- hold joint conferences
- hold joint demonstrations
- jointly publish papers and articles

## **ADMINISTRATION OF JOINT ACTIVITIES**

The options for hosting these joint activities are several.

- NIST
- DOO
- ITRC
- MAP/TOP
- others

ITRC was established to promote the partnering of government and industry in the pursuit of the basic objective that all three of the groups under discussion share today - the acceleration of the development and acceptance of information technology standards and the availability of a wide choice of tested and compliant products that meet user requirements.

ITRC is therefore recommended as the organization to administer this process. However, any reasonable alternative that effectively addresses the problems outlined in this paper would be acceptable.

## **RESULTS**

If this proposal is acted upon, the results would be:

- A dramatic acceleration of the availability of products that conform to information technology standards.

- Better choice of products for all users

- Improved reliability and compatibility

- Better functionality

- Lower cost

Elimination of the current confusion that government and industry are independently converging on different sets of standards.

**A more efficient process to assure compatible products that meet all user requirements**

**Elimination of duplicate committees**

**Better utilization of scarce resources to pursue user goals**

**Elimination of extra steps to 'harmonize' results of parallel processes**

**More cost-effective, timely approach to accomplish shared goals.**

**A collaboration that will encourage other United States users to join this process instead of starting up new activities which will create additional confusion.**

**A partnership that will allow the United States to more effectively leverage its interests in international standards development and allow us to more effectively compete in the world community.**





THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

FINAL REPORT  
CALS FY89 SOW TASK 4.1.4  
CGM REGISTRATION IN SUPPORT OF  
CALS REQUIREMENTS





## TABLE OF CONTENTS

1. INTRODUCTION . . . . .	1
1.1 Progress to date . . . . .	1
1.2 CGM Extension Philosophy . . . . .	2
2. LINES AND HATCHES . . . . .	3
2.1 General Approach to "Hatches" . . . . .	3
2.2 User-defined "Hatch" Style . . . . .	4
2.3 Other Required Graphical Elements . . . . .	5
2.4 Color Support . . . . .	6
3. TEXT . . . . .	7
3.1 Introduction . . . . .	7
3.1.1 Codes and Character Sets . . . . .	7
3.1.2 DIS 9541, Font Information Interchange, and ISO 10036, Procedures for Registration of Glyph and Glyph Collection Identifiers . . . . .	8
3.1.3 Character Sets and Fonts in the CGM . . . . .	11
3.2 Character Sets . . . . .	12
3.2.1 Character Sets in Commercial Practice . . . . .	12
3.2.2 A Phased Approach to Character Sets for CALS Use . . . . .	13
3.3 Character Set and Glyph Association . . . . .	13
3.4 Font List Contents . . . . .	14
3.4.1 Fonts in Commercial Practice . . . . .	14
3.4.2 A Phased Approach to Font Lists for CALS Use . . . . .	15
3.5 Registration of "Font" Names . . . . .	17
3.6 Recommended Approach to Font Naming in CALS . . . . .	18
3.7 CGM Text Model Extensions . . . . .	21
3.7.1 Basic Text Model . . . . .	21
3.7.2 Graphical Primitives for Text . . . . .	24
3.8 Font Substitution Support . . . . .	24
3.9 User-defined Fonts . . . . .	26
4. NAMED ITEMS AND SYMBOL LIBRARIES . . . . .	28
4.1 Requirements for Named Items and Symbol Libraries . . . . .	28
4.2 Alternative Approaches . . . . .	30
4.2.1 Naming . . . . .	30
4.2.2 Symbol Definition . . . . .	31
5. SPONSORSHIP OF REGISTRATION PROPOSALS . . . . .	33
APPENDIX A: CHARACTER SET EXAMPLES . . . . .	37
APPENDIX B: FONT EXAMPLES . . . . .	87
APPENDIX C: REGISTRATION PROPOSALS . . . . .	155





## 1. INTRODUCTION

This is the final report for FY89 in support of CALS SOW Task 4.1.4, which states: Continue to define CGM extensions needed to support CALS. Support identified extensions through the registration process.

This task was further subdivided into four subtasks as specified below:

### 1. Lines and Hatches:

Define requirements for lines, hatches and text font usage to support automated publications. Identification of requirements will be based on applicable MIL-STD's and existing commercial practice.

### 2. Text:

Develop and define a new text Graphical Display Primitive (GDP) and associated escapes (to implement attributes) that can fully implement the portions of the model of ISO DP 9541 which are appropriate for graphics standards. Extensions to the "CGM Environment" compatible with the ISO font description and transfer work shall also be developed.

### 3. Named Items and Symbol Libraries:

Develop extensions to CGM to define collections of graphical primitives as macros or symbols.

### 4. Sponsorship of Registration Proposals:

Negotiate any required changes with the X3H3 committee. Following ASC X3H3 approval, sponsor each proposal through the ISO registration process, responding as necessary to requests for changes and clarification. For previously proposed items, continue support through the registration process.

## 1.1 Progress to date

In the area of CGM lines and hatches, no additional requirements for specific lines or hatches were generated during this fiscal year. This report provides the rationale used to develop the previously defined user-defined line styles and defines the requirements for a user-defined hatch style. A registration proposal is included for a general fill that can meet the needs of a "user-defined hatch." This report also identifies the need for support for closed figures. Since Closed Figure support is

included in CGM Addendum 1, no additional proposals were developed, as those in Addendum 1 are suitable for CALS use.

In the area of text, this report provides extensive analysis and requirements development. A phased approach is taken, with both near term and future strategies being considered. The relationship of text font to character set is explained and the difficulty of registering new character sets is described. The naming of "fonts" is described and various approaches to what should be "named" are compared and the difficulties described. The architecture derived is a significant contribution towards achieving compatible usage of font resources among various presentation processes (including CGM-based computer graphics systems, SPDL, ODA, and SGML systems). No CGM extensions are proposed in the area.

In the area of named items and symbol libraries this report develops a set of requirements and provides extensive analysis of competing approaches. As with text, a phased approach is taken, with both near term and future strategies being considered. The approach taken is compatible, in the long run, with the approach to text. It will allow the eventual definition of font and symbol resources that are compatible as possible.

In the area of sponsorship of registration proposals the work done is described. There were a set of comments on previous proposals that were "referred back to the proposer," and changes to these proposals based on the comments have been made. At the Nashua X3H3 meeting (25-29 September 89) modifications to a previous set of proposals that allowed them to pass X3H3 ballot were agreed on. The final versions of these modifications are also included here.

## 1.2 CGM Extension Philosophy

There are several important assumptions that compose the philosophy of CGM extensions. These have been implicit in the CALS work to date, but should be formally documented to help others understand this work. These assumptions are:

- 1) Whenever possible, adopt extensions directly from other standards or de-facto standards that reflect predominant commercial practice. This includes adopting flawed and non-optional solutions where operability is not degraded and compatibility is enhanced.
- 2) Any CALS CGM should be interpretable by a non-CALS system in a predictable way.

- 3) Invention of new primitives and concepts should be avoided if an existing CGM element's meaning can be reasonably modified to meet CALS requirements. (This maximizes the extent to which non-CALS interpreters can interpret CALS CGM files.)
- 4) Make only the minimal extensions required for CALS rather than more general ones useful to other application areas.

## 2. LINES AND HATCHES

Based on review of MIL-STDs in hand and commercial practice, no additional specific line styles or hatch styles have been able to be identified as required for CALS, other than the user-defined hatch style as defined below.

Several "line-types" have been identified as needed in what appear to be a limited number of cases in drawings constructed according to various MIL-STDs. Since there are not enough examples of CALS document practice to judge the universality of many of these, the approach adopted has been to implement those of them that cannot be done with the existing user-defined line type proposal by using a closed figure construct. A further argument against registering additional lines and hatches at this time is that the arguments for them derive primarily from engineering drawing exchange, not technical illustration exchange format. The CGM is not yet an allowable exchange format for engineering drawings in CALS. For this reason, a set of additional graphical constructs are defined below that are needed CGM extensions to meet CALS requirements. These are adopted from CGM Addendum 1.

### 2.1 General Approach to "Hatches"

For reasons explained in previous CALS reports, NIST/NCSL has adopted an approach of meeting CALS requirements for treatment of the interiors of filled objects by slightly extending the concept of "hatch" to allow irregular fills. Although the ANSI X3H3 committee was convinced of the appropriateness of this approach, US delegates were not able to convince their ISO counterparts at the ISO level registration meeting held earlier this year. Consequently, several proposals for hatch styles were not accepted for registration. These were essential for CALS use. Fortunately, they are already part of the CALS CGM AP (Mil-D-28003).

No change in the approach is planned for several reasons:



- 1) The ISO objections were based on a narrow interpretation of the definition "hatch" as defined in GKS.
- 2) The only alternative to expanding the definition of hatch is to define a new "interior style" different from the presently allowed hollow, solid, pattern, hatch, and empty. This is difficult to do by registration alone since the list of interior styles cannot itself be extended through registration. Nonetheless, a set of registration proposals is being submitted with this report to allow a general fill. The requirements to be satisfied with "extended hatch" can be met, although not optionally, with this general fill. It is included to meet certain technical illustration requirements that cannot be met with predefined, registered hatch patterns.
- 3) In the long term new Application Programmer Interface standards (such as revisions to GKS and PHIGS) and CGM extensions work is expected to validate the need for a new alternative interior style treatment. This style could be added to the list of allowable interior styles, or the concept of hatch could be extended.

## 2.2 User-defined "Hatch" Style

It is evident that there are special circumstances where there will be areas whose interiors must be "filled" with a more or less regular graphical "pattern." The more common of these have been previously identified and proposed for registration. Less common and more specialized ones are needed nonetheless, especially in technical illustrations, and some mechanism must be found for accommodating them. Since resulting drawings are not revisable, these interior treatments cannot be "rendered" by drawing them with individual graphical primitives. This is because the primitive scale is the drawing. A further argument for their use is the additional compactness that results from describing a repeating graphical "pattern" only once.

It takes only a brief review of the hatch styles thus far proposed for registration to realize that no restrictions can be placed on the allowable graphical primitives that can define the "pattern" in such a filled area. In particular, allowing only traditional straight lines in such a user-defined hatch is not adequate. Similarly, raster-like patterns are inadequate. What is required is a concept similar to what Page Description Languages (PDLs) call "ink," whereby a "picture" can be drawn using any valid operators and repeated on a regular basis throughout the area to be filled. Consequently the Generalized Fill escape proposed with this report will allow any valid

picture to be used as its "fill pattern." Existing reference points and sizing information for fill patterns can be used to place and size the filling picture.

## 2.3 Other Required Graphical Elements

Several graphical elements are required both to allow user-defined edges and lines and to meet requirements identified for defining closed figures by composite curves. All of these can be based on elements defined in the CGI, so little controversy is expected over their definitions. During the last year, CGM Addendum 1, which incorporates these CGI facilities, has advanced far enough so that adopting the following elements into the CALS AP can be recommended rather than submitting duplicate proposals through the registration process:

- 1) BEGIN FIGURE
- 2) END FIGURE
- 3) CONNECTING EDGE
- 4) NEW REGION

These escapes are used to construct closed figures as follows. BEGIN FIGURE starts the accumulation. Subsequent graphical drawing primitives or GDPs (not the limited set allowed in the CGI proposal) are accumulated into the figure. This process is stopped by an END FIGURE escape. The CONNECTING EDGE escape controls whether edges are drawn connecting subsequent primitives that might not otherwise connect. The NEW REGION escape causes the sub-figure being accumulated to be closed and a new sub-figure to be started.

In addition NIST/NCSL does not perceive requirements for clipping to arbitrary regions within CALS. A general user-defined "fill" can create most, if not all, the graphical effects that clipping to arbitrary regions can achieve. In the remaining cases, (if any) the burden is best placed on the generator to perform the clipping, rather than on all interpreters. The generating system must already be able to do this kind of clipping since it must "preview" the CGM files or the local drawings they are derived from. There are further important arguments for a generalized "fill" over clipping to arbitrary boundaries:

- a) Interpreters that cannot clip to arbitrary boundaries would get a very wrong picture, with the objects being clipped spilling over onto other areas; in the generalized fill approach, the correct objects would still be drawn and appropriate defaults could cause it to still look about right.



- b) Usually it is not desirable to have the "insides" of arbitrary filled areas transform as the picture is revised; the "clipping to arbitrary boundaries approach" cannot prevent this.

## 2.4 Color Support

Some color models, such as CMYK, are device and printing process-dependent. (The amount of "K", usually "black", that is applied depends on printer ink characteristics and on mechanical or physical reasons. "K" is used to achieve highly saturated areas of black, in lieu of more expensive cyan, (C), yellow (Y), and magenta (M) inks, or to prevent paper from getting too wet.) Therefore they are not suitable for device-independent interchange. Others, such as CIE models, provide a degree of precision and device-independence that is pointless for CALS applications, and can be avoided (for CALS purposes) by simply using calibrated exchange based on an RGB model. Besides, most CALS technical illustrations will be monochrome or gray-level.

Industry practice today is to exercise precise color control only for final form documents and then only as represented in four color separates and only within a single system. The following typical list of steps shows the nature of the complex, device-dependent processing involved:

- 1) Producing three separate RGB bitmaps representing the picture.
- 2) Transforming these bitmaps through a complex, system-dependent process into CMYK bitmaps. These transforms correct for:
  - a) device calibration;
  - b) different color gamuts in the display and printing devices;
  - c) gray balance, gray component replacement, and production of a black separate.
- 3) Production of a Cromalin proof (hard-) copy from the separates to provide to the printer.

Therefore, support for additional colour models is not required.



### 3. TEXT

There are many issues to address in extending the CGM to support the text requirements for CALS. After providing necessary background information and reiterating the text requirements developed previously, these issues are defined and addressed in turn. The major issues are:

- 1) What character sets are required for CALS?
- 2) How should character sets be associated with the glyphs in a font?
- 3) What should the contents of a "font list" be?
- 4) What "names" should be registered for fonts?
- 5) What extensions, if any, are needed to the CGM text model to support use of DIS 9541 font resources?
- 6) To what extent should font substitution information be supported in the CGM?
- 7) Are user-defined fonts required? If so, what approach should be taken?

Subsequent paragraphs below address each of these issues in turn.

#### 3.1 Introduction

This subsection provides a condensed version of background information from standards in the areas of computer graphics, codes and character sets, and office systems. Knowledge of this material is necessary to understand the concepts presented elsewhere in this report. Throughout, standards are referred to by their ISO rather than ANSI or FIPS designators. Most of the indicated standards have, or will have, both ANSI and FIPS counterparts.

##### 3.1.1 Codes and Character Sets

The first standard concerning coded character sets was ISO 646-1973. It defined a basic 7 bit character set designed to be specialized for national use by assigning country-specific values to certain national symbols, such as the currency symbol. ANSI X3.4-1977 specialized ISO 646 for US use as the familiar "ASCII" character set. ISO 2022-1986 defined an extensible framework whereby both 7 and 8 bit character sets could be built from smaller components called "C" sets and "G" sets of 94 or 96 characters ("C" for control and "G" for graphic).

These C and G sets can be switched in and out of the active code table through a process known as code extension. In this technique, each C set or G set is identified by a unique escape sequence. Escape sequences can be registered in the International Register of Coded Character Sets to be Used with Escape Sequences or can have a privately-understood meaning. For example, the ASCII "G" set is invoked as the primary graphics character set (called the G0 set) by the sequence "ESC 02/08 04/02." Appendix A contains the descriptions of both the ASCII character sets and the Latin alphabet No. 1 character sets from this register.

More recently, ISO 6937-1983 has consolidated previous standards and defined general techniques for the use of coded character sets in text communication. Among the subsequent standards based on 6937 is ISO 8859 which defines 8-bit single byte coded graphic character sets. Part 1 of ISO 8859 defines Latin alphabet No. 1 by consolidating the two G sets from ISO 646 and the registered version of Latin alphabet No. 1. Appendix A contains a copy of the pages of ISO 8859-1 that define Latin alphabet No. 1.

The codes and character sets in all computer graphic standards today are based on the above standards. These standards and the coding techniques embodied in them were developed by ISO/IEC JTC1/SC2 and are based largely on telecommunication requirements. They are less than adequate for describing and coding typographic quality text.

ISO 8879 (SGML) takes a somewhat different approach to character sets. Beyond use of a subset of the basic "ASCII" characters (referred to by their decimal equivalents in Figure 1 of ISO 8879) additional "character entity sets" are defined in Appendix A. Also an ISO 646/ISO 2022 "multicode" structure allows the use of standard G0 and G1 national character sets.

### 3.1.2 DIS 9541, Font Information Interchange, and ISO 10036, Procedures for Registration of Glyph and Glyph Collection Identifiers

These standards define an alternative scheme for dealing with "codes and character sets." The first step in this process is to separate the "symbol" being represented from the "code" used to identify it in transfer. The term glyph is defined to replace the overused term character. A glyph is simply an identified abstract graphical symbol independent of any actual image. The term glyph collection is a precise substitute for the loosely defined term font. Glyphs and glyph collections are registered and assigned identifiers according to the procedures defined in ISO 10036. Appendix B contains a somewhat out-of-date sample from a prototype glyph register showing the same accented Latin



characters described in Latin alphabet No. 1. It also contains sample forms used for registering glyphs.

The font information interchange architecture and format defined in DIS 9541 is centered on an object called a font resource. A font resource is a collection of glyph representations together with descriptive information and font metrics which are relevant to the collection as a whole. A font resource consists of:

- 1) Descriptive attributes
- 2) Font metrics
- 3) Glyph descriptions
  - Glyph metrics
  - Glyph shapes

Some of the information in a font resource depends on the "path" along which the text progresses as it is written. This is called the writing mode in DIS 9541. Thus a single font resource may contain several sets of font metrics and glyph metrics, one for each supported writing mode.

The complete set of font information is not needed for all purposes, so the interchange format defined in DIS 9541 identifies named subsets that incorporate each major class of information. These subsets are:

- 1) Minimum Font Description Subset
- 2) Minimum Font Metric Subset
- 3) Minimum Glyph Description Subset
- 4) Minimum Glyph Metric Subset
- 5) Minimum Glyph Shape Subset

The set of information in each subset is:

- 1) Minimum Font Description Subset
  - o Font resource name (e.g. ISO9541/Helvetica/Bold)
  - o Source name
  - o Font family name (e.g. Helvetica)
  - o Posture (upright, oblique, back slanted oblique, italic, back slanted italic, or other)
  - o Weight (ultra light, extra light, light, semi light, medium, semi bold, bold, extra bold, or ultra bold)
  - o Proportionate width (ultra condensed, extra condensed, condensed, semi-condensed, medium, semi expanded, expanded, extra expanded, or ultra expanded)
  - o Design group (e.g. serif)
  - o Structure (solid, outline, inline, shadow, or patterned)



- o Design size (the body size at which the resource was designed to be used, in mm.)
  - o Minimum size
  - o Maximum size
- 2) Minimum Font Metric Subset (repeated for each writing mode)
  - o Writing mode (left to right, bottom to top, right to left, or top to bottom)
  - o Average lower case escapement
  - o Average capital escapement
  - o Tabular escapement
- 3) Minimum Glyph Description Subset (repeated for each glyph)
  - o Glyph structured name (see discussion below)
- 4) Minimum Glyph Metric Subset (repeated for each glyph, unless the font is fixed pitch)
  - o X position point
  - o Y position point
  - o X escapement point
  - o Y escapement point
- 5) Minimum Glyph Shape Subset (repeated for shape representation technology supported by the font resource)
  - o Glyph shape representation technology (bitmap, straight-line, outline, circular outline, conic outline, Bezier outline, or mixed)
  - o Glyph shape ( a "code-body" defining the shape; it is assumed that CGM elements, or primitives that can be easily translated to CGM elements will be used)

Certain additional information is included only in the complete font resource and is not in any subset. Important information in this category includes:

- o Device-independent font resource name (a font resource containing this one, together with other resources that use different shape description technologies)
- o Proprietary data indicator (e.g. copyright and trademark citations)
- o Typeface name (e.g. Helvetica Bold Italic)
- o Glyph complement (identification of the glyphs in the font)
- o Ligature table

- o Minimum feature size
- o Forward, left, backward, and right extents
- o etc.

Each glyph will have a structured name by which it will be referenced in DIS 9541. The structured name will be composed of the glyph description, as illustrated in the sample in Appendix B, preceded by other identifiers in a hierarchical fashion. For example, the "Grave A" that shows up in Latin alphabet No. 1 in position 00/04 and is called "CAPITAL LETTER A WITH GRAVE ACCENT" and with bit combination 12/00 in ISO 8859-1 might have a structured name of "ISO10036/ACCENTED LATIN CHARACTERS/GRAVE A".

One issue that has yet to be settled is the precise form of the "code-body" that will describe the glyph shapes. Several alternatives have been mentioned:

- 1) line segments
- 2) conics
- 3) Bezier curves
- 4) bitmaps

The ultimate usability of font resources by computer graphics standards hinges upon the adoption of a shape description technique that is readily translatable into CGM/CGI-like graphical primitives and attributes.

### 3.1.3 Character Sets and Fonts in the CGM

The metafile descriptor of each CGM file can contain a FONT LIST and a CHARACTER SET LIST. A font list is a set of strings that contain the names of all fonts used in the file. These names may be private or registered. The font list also associates an index with each font. The character set list declares the character sets that are used in text primitives. It also assigns an index to each character set. A CHARACTER CODING ANNOUNCER element informs the interpreter of the sort of code extension capabilities that may be required.

The CALS CGM AP (MIL-D-28003) restricts these values as follows:

FONT LIST: up to four names from a basic list of Hershey fonts (e.g.       HERSHEY:SIMPLEX\_ROMAN)

CHARACTER SET LIST: ASCII plus the right hand part of Latin Alphabet No. 1 (as defined in ISO 8859-1)

CHARACTER CODING ANNOUNCER: basic 7 bit and basic 8 bit.

Therefore, the relationship between font and character set in the CGM is an implicit one inferred through the use of ISO character coding standards.

### 3.2 Character Sets

This section derives a phased approach to character set usage in CALS CGM files, based upon the requirements and existing commercial practice.

#### 3.2.1 Character Sets in Commercial Practice

Today most vendors provide a proprietary character set that includes the 7-bit ASCII standard of ANSI X3.4 as a subset. Most 8-bit proprietary character sets include at least a part of the right hand part of Latin alphabet No. 1. Also, most proprietary character sets replace less frequently used ASCII and Latin characters with customized glyphs. Appendix A contains example vendor-specific character sets from the Apple Macintosh and the IBM PC that clearly illustrate this phenomenon.

In addition, a variety of specialized character sets are used in mathematics, technical illustrations, and engineering drawings. These special character sets often do not conform to the architecture of ISO 2022 and have never been registered in the International Register of Coded Character Sets. This means that no standardized escape sequences are available for their invocation in a CGM. Appendix B illustrates this by presenting extracts from IGES 4.0 that show the 14 "fonts" that are supported. Many of these "fonts" have custom character sets (glyph collections) defined with them.

SGML defines the following set of "character entity sets" all of which are allowed in CALS SGML files by MIL-M-28001A:

- 1) Added Latin 1
- 2) Added Latin 2
- 3) Greek Letters
- 4) Monotoniko Greek
- 5) Russian Cyrillic
- 6) Numeric and Special Graphic
- 7) Diacritical Marks
- 8) Publishing
- 9) Box and Line Drawing
- 10) General Technical
- 11) Greek Symbols
- 12) Alternative Greek Symbols
- 13) Added Math Symbols: Ordinary



- 14) Added Math Symbols: Binary Operators
- 15) Added Math Symbols: Relations
- 16) Added Math Symbols: Negated Relations
- 17) Added Math Symbols: Arrow Relations
- 18) Added Math Symbols: Delimiters

The SGML standard mentions that these groupings reflect the requirements of ISO 6937, but these particular character sets cannot be found in any ISO character set standard or in the International Register.

### 3.2.2 A Phased Approach to Character Sets for CALS Use

Many CALS requirements can be met by using the two character sets supported in the CALS CGM AP today: ASCII and the right hand part of Latin alphabet No. 1. Unfortunately, many technical illustrations derived from IGES files will contain special "characters" from the Symbol 1, Symbol 2, and Drafting fonts. There are many examples of technical illustrations that contain Greek letters or mathematical symbols. To meet these long term requirements the special SGML-defined character sets described above should be supported, as well as the IGES character sets described above.

There are three feasible approaches to meeting these requirements:

- 1) Propose the needed character sets for registration through ISO.
- 2) Add the needed character sets to the CALS CGM AP and designate private escape sequences to designate them.
- 3) Leave it up to each DoD program to define any additional character sets it needs.

The second approach is best in the near term: add at least three IGES character sets (Symbol 1, Symbol 2, and Drafting fonts) and the eighteen SGML character sets to the CALS CGM AP. The process of formal ISO registration of these character sets could simultaneously begin, as this would be a preferable long-term approach. (None of these character sets appear in the register today.)

### 3.3 Character Set and Glyph Association

Longer term, CALS requirements are best met by defining CGM extensions that de-couple the notion of character set from glyph. As the full DIS 9541 font resource model is implemented in the

CGM such a glyph-to-character set association will no longer be available as it is today due to the exclusive use of coding standards developed by ISO/IEC JTC1/SC2. With this report a new CGM element is being proposed that will allow the association of a set of codes with a set of structured glyph names and with an escape sequence. It is likely to be needed in the long run by CALS. Since it will take some time to achieve consensus in this area, it may be best to start the process now.

The requirements developed for this CGM extension are that it:

- 1) be independent of code length (allowing 8, 16, 32,... bit coded "character sets");
- 2) allow registration of commonly-used associations and their invocation by a simple identifier;
- 3) be compatible with DIS 9541 and ISO 10036 (e.g. use structured glyph names);
- 4) allow specification by changing only selected codes from a standard set;
- 5) allow citation of any ISO registered glyph collection in a straightforward and simple manner.
- 6) allow definition of a glyph collection and its association with a "character set" in a straightforward and simple manner.

The specific escape proposed does not meet all these requirements since additional technical work is needed in this area. It will however meet near-term requirements in an expeditious manner.

### 3.4 Font List Contents

This section addresses requirements for "calling out" fonts in the CGM. It surveys current vendor practice, the requirements imposed by DIS 9541 compatibility, and CALS requirements. Based on these it develops a phased approach to font list contents in CALS.

#### 3.4.1 Fonts in Commercial Practice

Commercially available computers and peripherals use a wide variety of different fonts. Appendix B lists the font capabilities of many commercially-available printers and plotters. It is readily apparent that a small core of fonts, or their generic equivalents, are widely implemented. These are:

- Helvetica
- Times
- Courier

In addition, it is apparent that a wide variety of fonts are available to meet special needs. The availability of many of these fonts is restricted to a single manufacturer.

Inspection of many IGES files indicates that most engineering drawing exchanges use either Symbol 1 or Symbol 2 fonts. These fonts/character sets are not widely supported outside of CAD systems.

Fonts in systems today are generally acquired from a small set of vendors. These include MonoType, Allied, Bitstream Inc., International Typeface Corporation (ITC), URW Corp., Adobe, and Compugraphic Corporation. While the master descriptions of a font are generally in "analog" form as drawings on paper (see the examples in Appendix B), they are usually translated into one of a variety of shape representation technologies, such as raster bitmaps and Bezier curves, for use within a system.

Many vendors provide publically available font metrics in file form. These files are not substantially different from the equivalent subsets of a DIS 9541 font resource. Appendix B contains a description of the font metric file format used by Adobe Systems.

#### 3.4.2 A Phased Approach to Font Lists for CALS Use

Most CALS requirements for fonts in technical illustrations, which is the most immediate CALS need, can be met with a very small set of government mandated fonts. These include:

- Helvetica, or a look-alike
- Times, or a look-alike
- Courier, or a look-alike
- mathematical symbol fonts

At the point when CALS wants engineering drawings transferred using the CGM, then at least the two IGES "symbol" fonts will have to be added.

Last fiscal year NIST/NCSL submitted the first of a series of registration proposals for adding enhanced text font capabilities to the CGM. The approach taken then was to match predominant commercial practice and minimize interpretation burdens on implementers by:

- 1) using font family names, like Helvetica and Times, in the font list;



- 2) set the other attributes of the font by escape functions (for example, boldness is turned on and off by an escape);

The rationale for this approach was:

- 1) It minimized the size of the font list and the number of names that must be either registered or agreed upon by profile or private agreement. For example, if font names rather than font family names are used, then registering names (or assigning them in a profile) would require the listing of at least 6 postures X 10 weights X 9 proportional widths X 5 structures = 2700 names for a single family such as Helvetica! (This assumes listing all possible variants of posture, weight, proportional width, and structure identified in DIS 9541. For example, there must be separate listings for Times Italic Bold Medium Solid and Times Upright Bold Medium Solid.)
- 2) It minimized the difficulty of parsing the names in a font list to recover attributes.
- 3) It recognized that most systems have an entire font family in all sizes and many attributes available; those systems (notably low cost laser printers) that do not have an entire font family cannot derive their font support requirements from the font list anyway, since their "font resources" are size dependent.

The next logical step beyond this approach is to allow structured font names (registered or private) in the font list. Such names can imply some attributes of a typeface ("font"), including such things as its weight (boldness) and posture (italicness). Unfortunately, it will be several years before the mechanisms of DIS 9541 are in place and a set of conforming font resources are available. There is even some doubt as to whether the major font foundries, such as ITC and Bitstream, will participate in defining font resources for their fonts. (They do not participate in the work within ISO/IEC JTC1/SC24 that is developing the necessary standards.) Therefore full DIS 9541 support, including use of registered names and vendor-supplied font resources, must be regarded as a long term goal.

The arguments above then lead into a consideration of what the next logical steps should be towards getting useful font information in CALS CGM files. The options are:

- 1) Continue to allow only the Hershey fonts as options (the current MIL-D-28003 approach);

- 2) Invent a set of "generic" font descriptions for use with CALS. Add these to MIL-D-28003; or
- 3) Use the typographic names of fonts as used in current commercial practice in the font list, where the allowable names could be:
  - a) registered;
  - b) listed in the CALS AP (MIL-D-28003); or
  - c) left up to individual DoD programs to specify.

In considering these options, it is concluded that:

- 1) Option 1 is rejected because:
  - a) The Hershey fonts do not represent current practice in document and illustration systems. In fact all these systems have used typographic fonts for some time and none support Hershey fonts. (See the data sheets in Appendix B.)
  - b) The quality achievable with Hershey fonts is not acceptable in commercial practice.
- 2) Option 2 is rejected because:

While generic font names may be used within a single document and illustration system, generic variants are sufficiently different so that interchange cannot be based on them. For example, the character sets supported, the kerning pairs (or lack of them), and even the escapement values are different among generic variants of the same font. ("Generic variant" in this sense means, for example, the creation of a "Swiss" font by a vendor to avoid licensing the trademarked name Helvetica.)

Therefore, only option 3 has the possibility of matching current commercial practice.

### 3.5 Registration of "Font" Names

The feasibility of registering the trademarked names of actual fonts for interim use with computer graphics standards has been explored. However, several difficulties with this approach have emerged that cannot be overcome. They are:

- 1) The one example of text font registration in the registration procedures includes glyph shape information. This information is not publicly available for "real" fonts. Therefore only the names



and the glyph-to-code correspondence could be listed in the proposals. Such proposals will be regarded as incomplete by some countries such as the UK, and are unlikely to be accepted.

- 2) All glyphs in a given character set may not be represented in a font. Some countries (notably the UK) have indicated they will not approve any registration proposals that do not code the complete character set. This problem is the same one that prevents registration of the Hershey fonts.
- 3) Within ANSI X3H3, many prominent companies have stated that they will oppose the registration of any font names at all. Their feeling is that only IS 10036 registered glyph collection names should be used.
- 4) Registering a complete set of typographic font resource names is combinatorially prohibitive, as there is typically one such resource for each combination of font attributes. The 2700 names per font family derived above is not too unrealistic. For example, Adobe and Bitstream sell over 240 types Helvetica fonts today, each with a different set of attributes!

Consequently, it would be counterproductive to attempt to register a set of font names for use by CALS. Further, it is very unlikely due to very dynamic marketplace conditions (the Adobe and Apple/Microsoft/IBM font interchange format "wars") that any major font vendors will seek to voluntarily use the mechanisms of IS 10036 any time soon. Therefore the only feasible approach may be to modify the CALS CGM AP to specify allowable "font" names, which would require that the changes (outlined in the next section below) to the CALS CGM AP should be considered during some future revision.

### 3.6 Recommended Approach to Font Naming in CALS

Based on the above rationale and trade studies, including a basic set of trademarked names in the CALS AP should be considered during a future revision, plus freely allowing individual DoD programs to specify other font names as needed to meet their needs. The reasons for this are:

- 1) It is not feasible to register real font names.
- 2) Individual programs with homogeneous equipment may wish to use more specialized fonts. This should not be prohibited unless there is an overwhelming DoD-wide interest in insuring that all illustrations and drawings are imageable on all systems with "equivalent"



results. This may not be a "requirement" since it is not only technically infeasible today, but would greatly increase the cost and reduce their quality significantly.

- 3) There would appear to be no legal problems with listing trademarked names in an AP, so long as:
  - a) the trademark is cited;
  - b) the fonts are widely implemented and available from multiple sources; (For example, Times and Helvetica are both registered trademarks of Allied Corporation; but Times and Helvetica versions are sold by multiple sources and available in many printers.)
  - c) the specification of alternate fonts by name is freely allowed in the profile;
  - d) generators and interpreters are free to use substitute fonts with "equivalent" metrics. (This would require that the font metrics be both released by the vendors and transmitted in some fashion with the CGM file so that a match could be made.)
- 4) Receiving systems with generic versions of trademarked font names can easily determine an appropriate substitution.

Therefore, the following approach for defining names in the CGM font list has been developed and should be considered for addition to the CALS CGM AP as a future revision:

- 1) allow interpreters and generators to map local fonts into "close equivalent" font names where portability is enhanced, as long as such a mechanism does not restrict competition and the font metrics are available to all;
- 2) allow compact font lists that list only partially descriptive font names, combined with the use of escape functions that state attribute selections allow optimal font substitution strategies;
- 3) explicitly list the following names as allowed in a font list, while recognizing that neither generators or interpreters need use fonts from the trademark owners:
  - a) Times
  - b) Helvetica
  - c) Courier

[It is not feasible to list either SGML or IGES-derived font names at this time since accompanying character sets have not yet been defined.]

- 4) continue to allow the use of the present Hershey fonts for upward compatibility;
- 5) define an explicit way of constructing names for font lists.

Based on the various trade studies in this report, including the one on font substitution below the following naming technique has been derived:

- 1) No "naming authority" name should be used.
- 2) Each name should begin with a font family name.
- 3) Additional portions of the names can specify as detailed of font attributes as wished and in any order; the attributes that can be specified are restricted to these from DIS 9541:
  - a) posture
  - b) weight
  - c) proportional width
  - d) structure
  - e) scores.

In each case, the attribute name must be one of those specified in DIS 9541, except in the case of scores, where the names "underscore", "overscore", and "throughscore" shall be used. [Use of the DIS 9541 names is needlessly complex and non-intuitive since a left-to-right writing mode is assumed.]

- 4) Portions of names should be separated by the "standard" separator symbol used in OSI and office systems naming. This is the solidus glyph (/).
- 5) Only characters (and codes) in ASCII (X3.4-1977) should be used in names.
- 6) Those attributes that are explicitly listed in the font list shall not be substituted by the receiving system.
- 7) Those attributes that are selected by the registered Escape functions (posture, weight, proportionate weight, structure, and scores) can be substituted with the closest available equivalent value. The closest available equivalent is a value that is closest to the

selected one in the ordered list of attributes in DIS 9541. For example, both medium and ultra bold are allowable substitutes for bold, if semi-bold and extra bold are not available. If the selected value is available, it shall be used. [Both generators and interpreters are always free to substitute the "font family name" by an equivalent.]

Here are some examples of how this would work:

- 1) The following names would all be allowable in a Font List:

Helvetica  
New Century Schoolbook  
Times/bold/italic  
Times/italic/bold  
Helvetica/italic/normal/medium  
Courier/underscore

- 2) If a bold Helvetica font is **required** then the Font List should contain: "Helvetica/bold".

If substitution is allowable, the font list should contain: "Helvetica" and the Select Typeface Weight (bold) escape function should be used to indicate the preference for bold.

It is left for further study whether there is a meaningful way to include typeface design grouping in a font naming scheme. In this regard it can be pointed out that terms used colloquially in the computer graphics community, and indeed in many font lists, are not used correctly. For example, Serif and Sans-serif are not major typeface design groups, but are in fact subgroups that occur in many design groups. For example, the familiar Times Roman font family is classified as a "Rounded Legibility" font (It was designed in 1931 for the Times of London to squeeze more type onto a page in a "legible" way!). Most graphics experts would call it a "serif" font! The familiar Helvetica font, on the other hand, is a "Sans serif Gothic Neo-grotesque" font.

### 3.7 CGM Text Model Extensions

This section contrasts the CGM text model with the information available in a DIS 9541 font resource and with the requirements of commercial typographic practice.

#### 3.7.1 Basic Text Model



The CGM standard recognizes the following positioning points for text alignment:

- |           |                           |
|-----------|---------------------------|
| 1) top    | 6) left                   |
| 2) cap    | 7) right                  |
| 3) half   | 8) centre                 |
| 4) base   | 9) continuous vertical    |
| 5) bottom | 10) continuous horizontal |

These points are shown in Figure 1.

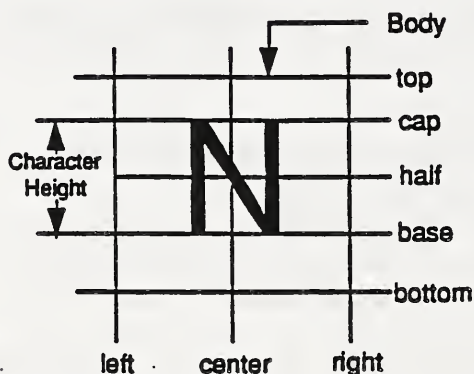


Figure 1. CGM "font description coordinate system"

The CGM standard recognizes four text paths:

- |          |         |
|----------|---------|
| 1) right | 3) up   |
| 2) left  | 4) down |

The "font model" of DIS 9541 uses the term "alignment line" rather than baseline, since the latter only has meaning in horizontal writing directions. A different alignment line can be specified for each writing mode. This is illustrated in Figure 2.

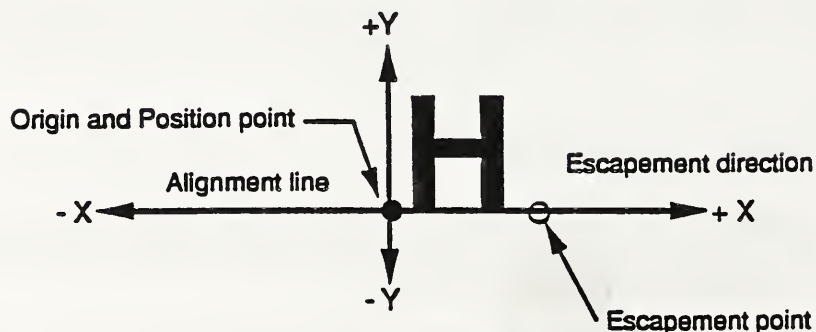


Figure 2. DIS 9541 text model.

The four supported writing modes in DIS 9541 (see section above) correspond directly to CGM text paths. Data to implement CGM baseline and centre alignment can be derived from the alignment lines in the various writing modes. In addition, the top, base, left and right alignment points can be derived from the Maximum Forward, Left, Backward, and Right Extents in the font resource. (The correspondence is writing-mode dependent.)

There is no "capline" information in a font resource, but the needed information can be derived from the Left Extent of the Capital H character in left to right writing mode. (This is the actual measurement that is used in typographic practice for Latin alphabets.) There is no "halfline" information in a font resource, but the needed information can be derived from the left extent of the lower case "x" in left to right writing mode. (This is the actual measurement that is used in typographic practice for Latin alphabets.) Information to implement continuous alignments can be similarly derived from the positioning point extent, and glyph shape information in a font resource. Table 1 below summarizes this information:

TABLE 1. Deriving CGM Text Attributes from a Font Resource

<u>9541 Font Information</u>	<u>CGM Alignment</u>
left to right mode, alignment line	baseline
right to left mode, alignment line	baseline
top to bottom mode, alignment line	centre
bottom to top mode, alignment line	centre
writing mode dependent extent data	top
writing mode dependent extent data	bottom
writing mode dependent extent data	right
writing mode dependent extent data	left
left to right mode, left extent of "H"	cap
left to right mode, left extent of "x"	half
positioning point, extents, glyph shape	continuous horizontal
positioning point, extents, glyph shape	continuous vertical

No revisions are necessary to the basic text model by registration to make the DIS 9541 information usable with the CGM. CGM addenda work should consider a closer alignment of text and font terminology at the next revision of the standard.

Two CGM text attributes do not correspond to typographic practice (for revisable documents) are:

- 1) CHARACTER EXPANSION FACTOR
- 2) CHARACTER SPACING

The use of these attributes should be deprecated by the CALS CGM AP since they conflict with typographic layout practice and hinder font substitution.

Finally, MIL-STD-1840A and/or the CALS Handbook will need to be modified at some future point to allow the inclusion of font resources as a file type, or at least insure that the font metrics are made available so that a "close approximation" to the actual, proprietary fonts may be realized.

### 3.7.2 Graphical Primitives for Text

As reported previously, the CGM text model and primitives are not suitable for CALS use without modification, since they do not support modern typographic practice. Once CGM Addendum 3 functionality has been adopted this problem will be resolved.

New text primitives are not needed in the CGM, since an existing primitive can be easily modified to meet requirements. (Adding new primitives by registration as GDPs would severely impact the portability of the resulting CGM files, since any interpreter that did not understand the GDPs would get no text. At least an interpreter that fails to understand the Set Fully Justified Text escape would only layout and/or size text incorrectly.)

Additional control over the line layout algorithm that is applied to position individual text characters into a string that fits a given restriction box are also needed. The problem is that individual systems and printers apply different, proprietary algorithms to fit text into a given space. These algorithms may modify:

- 1) the space between characters;
- 2) the space between words;
- 3) the shape (and consequently the extent) of individual characters.

The possibility of adding an additional escape, or modifying the one already proposed, to allow specification of the layout algorithm has been investigated. It has been concluded that it is adequate at present to add a suggested default layout algorithm to the description of the Set Fully Justified Text escape. This has been done.

### 3.8 Font Substitution Support

Font substitution is a subject of considerable importance in open interchange of graphical information. Since the generating and interpreting systems may not have "equivalent" fonts installed or



available, sufficient information should be provided to allow an intelligent selection of alternative fonts.

There are two approaches to font substitution information:

- 1) The receiving system derives the data from information in the font list. In current practice this is done by using either simple, "generic" names (such as "SERIF/BOLD), or vendor-specific names (such as "Helvetica"), that can be parsed to derive font characteristics. Once DIS 9541 is widely implemented, equivalent information will be available from the Minimum Font Description Subset. (Widespread implementation of DIS 9541 is still many years away.)
- 2) The CGM can be extended to carry the data in the Minimum Font Description Subset. This could be done by adding one or more escape functions to the CGM.

The first approach is probably the best one because:

- 1) Information in a font resource should be used by the CGM, not duplicated within the CGM. Duplication of this information within the CGM can lead to needless incompatibilities.
- 2) Even if the Minimum Font Description Subset is not available to a CGM interpreter, the name of the font resource alone is expected to carry enough data to allow substitution in most cases. (An example name would be Helvetica/Bold/Italic.)
- 3) The Font Resource Name has already been selected as the name that is carried in the "font list", (in the long run) so the name will be readily available for use in deriving substitute fonts.
- 4) The set of fonts that will be used most often in "portable" interchange will actually be quite small and will likely be restricted by application profiles. The information in the Minimum Font Description Subset is likely to be widely available for these fonts. In a CALS environment this subset could be included on MIL-STD-1840A tape deliverables.
- 5) In cases where special Font Resources are required, they can be interchanged along with the CGM files to make at least the information in the Minimum Font Description Subset available.
- 6) The first approach is consistent with the short term approach, where "font names" specified in the CALS CGM

AP are used in the font list. These names are sufficiently structured that font substitution information can be derived from them.

Font substitution will, in all likelihood, play only a minor role in initial CALS data exchanges (that is, the delivery of documents between parties for the first time). This is because of the many difficulties pointed out above in achieving high-quality results with non-identical fonts. In commercial practice today where high-quality results are needed, exchanges are based on the use of fonts with identical metrics, often from the same vendor. Contracts calling for CALS deliverables should place sufficient restrictions on the fonts and character sets used to make font substitution a minor concern.

Continued usability of CALS deliverables requires that documents be reproducible as easily as possible on a variety of equipment. This is necessary when suppliers change or when manufacturers go out of business. Consequently, there is a continuing requirement to indicate font substitution information within a CGM file. The registration proposals proposed to date go a long way towards meeting these requirements, but further work and study will be necessary in this area.

### 3.9 User-defined Fonts

There are several circumstances where user-defined fonts may be advantageous:

- 1) When a specialized drawing containing mathematics or drafting-derived material (where special symbols are used routinely) is transferred to a system without necessary font support.
- 2) In high-quality graphics arts where customization of fonts is used to achieve special effects.
- 3) In transfers to CGM interpreters where adequate font support does not exist.

None of these requirements exist for routine CALS data exchanges. In those rare cases where a special symbol is needed (for example, as part of a label for an illustration), that symbol can be drawn in the CGM file with geometric primitives. Consequently user-defined fonts are not viewed as a required CGM extension for CALS use. As previously stated, contracts calling for CALS deliverables should place sufficient restrictions on the fonts and character sets used to make user-defined fonts unnecessary. Further, as mentioned above, needed character sets, glyphs, and glyph collections should be defined and proposed for registration.

Nonetheless, for completeness, alternative approaches to implementing user-defined fonts are described below. There are two feasible approaches:

- 1) Use the Font Resource facilities of DIS 9541. The data in a font resource should be easily interpretable by most computer graphics systems, since it will consist of simple graphical primitives.
- 2) Develop CGM extensions for defining fonts. This could be done by adding a set of escape functions to "collect" the primitives that define a glyph into a glyph description and to define the glyph metrics.

The choice between these alternatives is clear: the Font Resource facilities of DIS 9541 should be used. A standardized mechanism in DIS 9541 for defining fonts is being added but is still several years away. To provide a duplicate facility within the CGM would be counterproductive and hinder compatibility and interoperability of office systems and computer graphics standards. Further, there are complex issues to be resolved dealing with "hints" necessary to allow outline fonts to be rendered at different resolutions. Hinting technologies are poorly understood in the computer graphics community today.

There is no justification for repeating information that is readily available within a font resource inside a CGM file. Therefore, whenever a user-defined font is required, a Font Resource should be developed for it. This resource could then be named in the font list and transferred with the CGM file for use by the receiving system.



#### 4. NAMED ITEMS AND SYMBOL LIBRARIES

The final task was to develop an approach to handling named objects and symbol libraries within the CGM framework. First, requirements in this area are reviewed and clarified, alternative approaches defined, and a design approach derived. Then a solution is presented that requires additions to the CALS CGM AP, MIL-STD-1840A, and the use of CGM Addendum 1 features, but only one new registration proposal.

##### 4.1 Requirements for Named Items and Symbol Libraries

The requirements for "macro objects" and symbol libraries derive from two principal sources:

- 1) the desire to reduce the size of an individual CGM file by including repeated information only once;
- 2) the desire to reduce the size of files by not retransmitting information in local libraries;
- 3) the desire to achieve uniformity by incorporating standardized objects such as symbols and logos into files by reference.

In addition, requirements for named objects derive from a desire to identify "compound objects" whose attributes can be changed as a whole rather than dealing with each individual primitive in the object.

Analyzing the above requirements in the context of CALS near term requirements (technical illustrations transferred on MIL-STD-1840A tapes) it is clear that file size is a lesser concern for two reasons:

- 1) file transfers will be on tape or optical disk, so file size is less important (although the local staging storage needed in going to and from tape is still an issue);
- 2) technical illustrations (as opposed to engineering drawings) contain few opportunities for compaction by extraction of repeated information.

Further, most CGM generators and interpreters cannot take advantage of a "macro" facility. Thus a phased approach to named objects and symbol libraries in CALS should focus on externally referenced symbols initially, then on providing compaction within individual files by an internal "macro" or "global segment" facility.

One issue that arises immediately is how external symbols can be named, referenced, and transferred for CALS use. Assuming initially that an 1840A tape containing all the symbols needed to produce a document is desired, there are several problems:

- 1) MIL-STD 1840A formats did not envision the use of external references within graphics files and does not provide a mechanism for CGM files to refer to other CGM files on a standard tape deliverable. This can be fixed by modifying the allowable contents of CGM "header files" to include the names of relevant CGM files.
- 2) Naming of external files should be consistent with their callouts in SGML files.

Finally, the requirements for eventual registration and/or standardization suggest that naming be compatible with the "structured name" concepts used in office system and OSI standards. In particular, this means that:

- 1) names should begin with a designator for a naming authority, whether public or private;
- 2) the remainder of the name should then be constructed in a hierarchical fashion to produce complete names.

For example, an acceptable structured name might be: "US DoD CALS/Electronic Symbols/PNP Transistor". Unfortunately, MIL-STD-1840A only uses 4 character designators (such as D001) without any indication of naming authority. Further, "compound names" are constructed by simple concatenation with no separators for parts of names. (This is adequate as long as names are fixed length!) For example, a CGM file might be named D001C001.

There is a longer-term requirement for the development of standardized libraries of named objects and for the registration of these names. It does not appear possible to use a subset of the DIS 9541 font resource objects and ISO 10036 glyph name registration techniques for these purposes. This is due to the fact that the graphical objects to be described will require more complex primitives than those required to describe glyph shapes. (The present intention is that glyph shape technologies will allow several different techniques, each of which is restricted to use only very simple classes of graphical primitives - such as lines, conics, or Bezier curves.) Perhaps a different registration authority than the one used for fonts would be necessary.



## 4.2 Alternative Approaches

### 4.2.1 Naming

The alternatives are:

- 1) use "semi-structured" strings as names;
- 2) use structured (registerable) names in the OSI sense.

Since no formal mechanism is in place for registering symbol names (or the "names" of any other graphical objects), the best approach is to simply use "strings" for names in any proposals and further restrict their contents in the CALS CGM AP. The use of structured names is not yet standard practice within the computer graphics community and attempts to impose a structure through registration rather than standardization will only delay the approval of proposals.

Consequently, names should be described as simple "strings" in registration proposals. A future revision to the CALS AP should further restrict the names of external symbols as described below, under the assumptions that:

- 1) all external symbols must be "deliverable" on a standard 1840A tape;
- 2) symbols are stored as global segments within a CGM file, as described below).

Thus, the derived naming technique should be:

- 1) no "naming authority" name should be used;
- 2) each name should begin with the file name of the CGM file that contains the symbol;
- 3) the local symbol name of the symbol within that file should come next (this means an character representation of the integer segment name);
- 4) portions of names should be separated by the "standard" separator symbol used in OSI and office systems naming. This is the solidus glyph (/).
- 5) only characters (and codes) in ASCII (X3.4-1977) should be used in names.

For example, the symbol for a PNP transistor might be stored in segment 10 in file D001C003. It would be named "D001C003/10".



#### 4.2.2 Symbol Definition

The following set of requirements for a single symbol definition facility for the CGM that can be used internally or externally has been developed:

- 1) Symbols should be able to be referred to by name.
- 2) When an "internal" (global) symbol is not present in a CGM, there should be a way to inform an interpreter that it is an "external" symbol. The external context to be searched is not standardized, but could be set by profile or other standard. For example, it might be limited to the same MIL-STD-1840A deliverable tape.
- 3) A symbol should be able to be the entire contents of a CGM file, and the entire contents of a picture should be an allowable symbol. [NOTE: The next revision of the CALS CGM AP will not deal with this, but it will be an interesting capability and a requirement once the CALS CGM AP is allowed for use in transferring engineering drawings.]
- 4) The metafile and picture attribute associated with a symbol need not be the same as those of the metafile that references it. In particular, attributes such as colour precision, VDC type and precision, VDC extent, colour specification mode, colour table, and integer precision may all be different. This is essential if the symbol library generation process is to be decoupled from the metafile interpretation process.
- 5) Symbols should be insertable by specifying (in effect) position, scaling, and rotation.
- 6) An arbitrary position point (in local VDC) should be associated with each symbol.
- 7) All attributes not specifically specified within a symbol definition should be inherited from the defining context.
- 8) No attribute settings within a symbol should change continuing interpretation of the including file.
- 9) Symbol names should be of unrestricted length to allow the convenient construction of structured names and external references.

After reviewing the features of CGM Addendum 1 in light of these requirements (with the view of using CGM segments as symbols) it appears that:

- 1) The "inheritance filter" concepts in Addendum 1 adequately addresses attribute inheritance at both the level of the CGM and of APIs. It is much more complex than required for CALS purposes and, if fully required, will place an extreme burden on implementers of CALS CGM interpreters.
- 2) The restrictions that segment names to be "realized" as integers is most unfortunate, but actually the problem is more syntax than fundamental semantics. The Addendum 1 mechanism gives a very compact and efficient internal name for a segment. The difficulty can be overcome by adding a Segment List element similar in scope and purpose to a Font List.

Nonetheless, the Addendum 1 features are usable for CALS purposes. Consequently, the symbol library elements developed under this task will not be proposed for registration, since compatibility with (nearly) standardized elements is more important than the ease of use of these elements.

Therefore, symbols that can be externally referenced should be stored as global segments in some CGM file. This could be called out in a local file by name using the Segment List element being proposed for registration. In addition, internal symbols could be contained in segments within any picture body, but they could not be externally referenced.

## 5. SPONSORSHIP OF REGISTRATION PROPOSALS

There continue to be long delays in getting registration proposals approved through the ANSI/ISO process. Most of these delays are due to demands by X3H3 members that registered items have the same level of "generality" as standardized items. This means that attempts to register limited and CALS specific items have not been successful, and that more work has been required, and will continue to be required to modify proposals than originally anticipated.

Despite these difficulties the use of registration to make computer graphics standards suitable for CALS use has been successful. Virtually without exception, NIST/NCSL proposals have been adopted by X3H3 without any significant technical change. Comments received on ballots are largely editorial in nature. A few comments ask for an additional level of specification of error handling and default behavior than is in common commercial practice. Such requests are easily accommodated.

The table below contains the current status of those registration proposals worked on this fiscal year. These proposals are divided into four classes, and appear in their entirety in Appendix C:

- 1) Proposals approved by X3H3 in September 1989 in Letter Ballot #80. Final text ready to forward to ISO is included for these.
- 2) Revised proposals from Letter Ballot #76. This text is ready for review and then re-ballotting. A significant amount of technical work has been done to align the proposals with DIS 9541 and to coordinate them with the CALS AP revisions being developed by another contractor.
- 3) Partially revised proposals, to which a common set of editorial corrections have been applied, but which need additional technical work prior to ballot.
- 4) New proposals submitted for the first time this year.



Table 2. Status of Registration Proposals

Class of Item	Name	Status
(Final revised text for these proposals is provided with this report.)		
Linetype	User-specified dash pattern	Approved by X3H3; final text with this report
Escape	Set dash	Approved by X3H3; final text with this report
Escape	Set line miter limit	Approved by X3H3; final text with this report
Escape	Set line cap	Approved by X3H3; final text with this report
Escape	Set line join	Approved by X3H3; final text with this report
Escape	Set conic arc transformation matrix	Approved by X3H3; final text with this report
GDP	Conic arc	Approved by X3H3; final text with this report
GDP	Parametric spline curve	Approved by X3H3; final text with this report
GDP	Rational B-spline curve	Approved by X3H3; final text with this report
GDP	Cubic Bezier curve	Approved by X3H3; final text with this report
(The following proposals have been revised based upon ballot comments are are ready for a second round of ballot comments.)		
Escape	Set edge miter limit	minor revisions included with this report; ready for review and ballot
Escape	Set edge cap	minor revisions included with this report; ready for review and ballot
Escape	Set edge join	minor revisions included with this report; ready for review and ballot

Table 2. Status of Registration Proposals-continued

Class of Item	Name	Status
Escape	Select Typeface Posture	revised and renamed proposal #46; ready for review and ballot
Escape	Select Typeface Structure	revised and renamed proposal #49; ready for review and ballot
Escape	Select Typeface Scores	revised and renamed proposal #51; ready for review and ballot
Escape	Select Typeface Weight	revised and renamed proposal #52; ready for review and ballot
Escape	Set Fully Justified Text	revised and renamed proposal #53; ready for review and ballot
Escape	Select Typeface Proportionate Width	revised and renamed proposals #54 and #55 combined; ready for review and ballot
(The following proposals have been partially revised to respond to comments of an editorial nature, but additional technical work will be necessary before they can be re-balloted.)		
GDP	Pel array	revised text with this report; additional technical work needed before another round of balloting within X3H3
Escape	Set direct colour response	revised text with this report; additional technical work needed before another round of balloting within X3H3
Escape	Set indexed colour response	revised text with this report; additional technical work needed before another round of balloting within X3H3

Table 2. Status of Registration Proposals-continued

Class of Item	Name	Status
(The following proposals are new with this report.)		
Escape	Select General Fill	ready for review and X3H3 ballot
Escape	Glyph Association	ready for review and X3H3 ballot
Escape	Segment List	ready for review and X3H3 ballot



APPENDIX A  
CHARACTER SET EXAMPLES



## ASCII CHARACTER SET





TYPE	GRAPHIC CHARACTER SET JEU DE CARACTERES GRAPHIQUES	REGISTRATION NUMBER NUMERO D'ENREGISTREMENT	006
REGISTRATION DATE DATE D'ENREGISTREMENT  1975/12/01	ESCAPE SEQUENCE  SEQUENCE D'ECHAPPEMENT	G <sub>0</sub> set / jeu G <sub>0</sub>	ESC 2/8 4/2
		G <sub>1</sub> set / jeu G <sub>1</sub>	ESC 2/9 4/2
		MULTIBYTE SET JEU MULTIPLIET	

NAME/NOM

ASCII Graphic character set

Jeu de caractères graphiques ASCII

#### DESCRIPTION

A set of 94 graphic characters which, when invoked in columns 2-7 of a 7 bit code table or an 8-bit code table would be positioned as shown on page 24. While alternate meanings are stated no substitution of graphic is intended.

Jeu de 94 caractères graphiques qui, lorsqu'ils sont appelés dans les colonnes 2 à 7 d'un tableau de code à 7 ou 8 éléments doivent être positionnés comme indiqué en page 24. Bien que des significations de remplacement soient admises, des substitutions de caractères graphiques ne sont pas prévues.

#### SPONSOR/ORGANISME DE PARRAINAGE

American National Standards Institute 1430 Broadway NEW YORK, N.Y. 100 18 U.S.A.

Organisme National de Normalisation des Etats-Unis

#### ORIGIN (USER)/ORIGINE (UTILISATEUR)

American National Standard X3.4 - 1968

Norme internationale américaine X3.4 - 1968



#### FIELD OF UTILIZATION/DOMAIN D'APPLICATION

Coded information interchange

Echange d'information codée

# ASCII graphic set

Coded representation when invoked in columns 2-7 of code table

				b.	0	0	1	1	1	1	
				b.	1	1	0	0	1	1	
				b.	0	1	0	1	0	1	
					2	3	4	5	6	7	
b.	b.	b.	b.								
0	0	0	0	0		0	@	P	`	p	
0	0	0	1	1	!	1	A	Q	a	q	
0	0	1	0	2	"	2	B	R	b	r	
0	0	1	1	3	#	3	C	S	c	s	
0	1	0	0	4	\$	4	D	T	d	t	
0	1	0	1	5	%	5	E	U	e	u	
0	1	1	0	6	&	6	F	V	f	v	
0	1	1	1	7	'	7	G	W	g	w	
1	0	0	0	8	(	8	H	X	h	x	
1	0	0	1	9	)	9	I	Y	i	y	
1	0	1	0	10	*	:	J	Z	j	z	
1	0	1	1	11	+	;	K	[	k	{	
1	1	0	0	12	,	<	L	\	l		
1	1	0	1	13	-	=	M	]	m	}	
1	1	1	0	14	.	>	N	^	n	~	
1	1	1	1	15	/	?	O	_	o		



ASCII graphic set

LEGENDS		
POSITION WHEN INVOKED IN COLUMNS 2-7	GRAPHIC	NAME OR MEANING (ADDITIONAL NAMES IN PARENTHESIS ARE ALTERNATE MEANINGS DETERMINED BY SYNTAX OR USAGE AND ARE NOT SUBSTITUTION ALTERNATIVES).
2/1	!	Exclamation mark
2/2	"	Quotation mark(diaeresis)
2/3	#	Number sign
2/4	\$	Dollar sign
2/5	%	Per cent
2/6	&	Ampersand
2/7	'	Apostrophe (closing single quotation mark , acute accent)
2/8	(	Left parenthesis
2/9	)	Right parenthesis
2/10	*	Asterisk
2/11	+	Plus
2/12	,	Comma (cedilla)
2/13	-	Hyphen (minus)
2/14	.	Full stop (period, decimal point)
2/15	/	Solidus (slant)
3/0	0	Digit zero
3/1	1	Digit one
3/2	2	Digit two
3/3	3	Digit three
3/4	4	Digit four
3/5	5	Digit five
3/6	6	Digit six
3/7	7	Digit seven
3/8	8	Digit eight
3/9	9	Digit nine

ASCII graphic set

LEGENDS		
POSITION WHEN INVOKED IN COLUMNS 2-7	GRAPHIC	NAME OR MEANING
3/10	:	Colon
3/11	;	Semi-colon
3/12	<	Less than sign
3/13	=	Equal sign
3/14	>	Greater than sign
3/15	?	Question mark
4/0	@	Commercial at
4/1	A	Capital letter A
4/2	B	Capital letter B
4/3	C	Capital letter C
4/4	D	Capital letter D
4/5	E	Capital letter E
4/6	F	Capital letter F
4/7	G	Capital letter G
4/8	H	Capital letter H
4/9	I	Capital letter I
4/10	J	Capital letter J
4/11	K	Capital letter K
4/12	L	Capital letter L
4/13	M	Capital letter M
4/14	N	Capital letter N
4/15	O	Capital letter O
5/0	P	Capital letter P
5/1	Q	Capital letter Q
5/2	R	Capital letter R

ASCII graphic set

LEGENDS		
POSITION WHEN INVOKED IN COLUMNS 2-7	GRAPHIC	NAME OR MEANING
5/3	S	Capital letter S
5/4	T	Capital letter T
5/5	U	Capital letter U
5/6	V	Capital letter V
5/7	W	Capital letter W
5/8	X	Capital letter X
5/9	Y	Capital letter Y
5/10	Z	Capital letter Z
5/11	[	Left square bracket
5/12	\	Reverse solidus (Reverse slant)
5/13	]	Right square bracket
5/14	^	Upward arrow head    circumflex accent
5/15	_	Underline
6/0	`	Grave accent (opening single quotation mark)
6/1	a	Small letter a
6/2	b	Small letter b
6/3	c	Small letter c
6/4	d	Small letter d
6/5	e	Small letter e
6/6	f	Small letter f
6/7	g	Small letter g
6/8	h	Small letter h
6/9	i	Small letter i
6/10	j	Small letter j
6/11	k	Small letter k
6/12	l	Small letter l



# ASCII graphic set

LEGENDS		
POSITION WHEN INVOKED IN COLUMNS 2-7	GRAPHIC	NAME OR MEANING
6/13	m	Small letter m
6/14	n	Small letter n
6/15	o	Small letter o
7/0	p	Small letter p
7/1	q	Small letter q
7/2	r	Small letter r
7/3	s	Small letter s
7/4	t	Small letter t
7/5	u	Small letter u
7/6	v	Small letter v
7/7	w	Small letter w
7/8	x	Small letter x
7/9	y	Small letter y
7/10	z	Small letter z
7/11	[	Left curly bracket (left brace)
7/12		Vertical line
7/13	]	Right curly bracket (right brace)
7/14	~	Tilde (overline, general accent)

(RIGHT HAND) LATIN I  
CHARACTER SET





<b>TYPE:</b> 96-Character Graphic Character Set	<b>REGISTRATION NUMBER:</b> 100 <b>DATE OF REGISTRATION:</b> Feb.1,1986												
<b>ESCAPE SEQUENCE:</b> <table> <tr><td>G0:</td><td>-</td></tr> <tr><td>G1:</td><td>ESC. 2/13 4/1</td></tr> <tr><td>G2:</td><td>ESC 2/14 4/1</td></tr> <tr><td>G3:</td><td>ESC 2/15 4/1</td></tr> <tr><td>C0:</td><td>-</td></tr> <tr><td>C1:</td><td>-</td></tr> </table>		G0:	-	G1:	ESC. 2/13 4/1	G2:	ESC 2/14 4/1	G3:	ESC 2/15 4/1	C0:	-	C1:	-
G0:	-												
G1:	ESC. 2/13 4/1												
G2:	ESC 2/14 4/1												
G3:	ESC 2/15 4/1												
C0:	-												
C1:	-												
<b>NAME</b> Right-Hand Part of Latin Alphabet Nr.1													
<b>DESCRIPTION</b> A 96-character set intended for allocation to columns 10 to 15 of the 8-bit code called Latin Alphabet Nr.1 in International Standard ISO 8859/1.													
<b>SPONSOR</b> <ul style="list-style-type: none"> <li>- ISO/TC97/SC2/WG-7</li> <li>- ECMA, EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION</li> </ul>													
<b>ORIGIN</b> <ul style="list-style-type: none"> <li>- ISO 8859/1</li> <li>- Standard ECMA-94</li> </ul>													
<b>FIELD OF UTILISATION</b> <p>This set of graphic characters is intended for use in data processing and text applications and may also be used for information interchange.</p> <p>The set contains graphic characters used for general purpose applications in typical office environments in the following languages :</p> <p>Danish, Dutch, English, Faeroese, Finnish, French, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish and Swedish.</p>													

				b.					0	0	0	0	1	1	1	1
				b.					0	0	1	1	0	0	1	1
				b.					0	1	0	1	0	1	0	1
									0	1	2	3	4	5	6	7
b.	b.	b.	b.													
0	0	0	0	0			NBSP	°	À	Ð	à	ð				
0	0	0	1	1			ı	±	Á	Ñ	á	ñ				
0	0	1	0	2			¢	<sup>2</sup>	Â	Ò	â	ò				
0	0	1	1	3			£	<sup>3</sup>	Ã	Ó	ã	ó				
0	1	0	0	4			¤	'	Ä	Ô	ä	ô				
0	1	0	1	5			¥	μ	Å	Õ	å	õ				
0	1	1	0	6			¦	¶	Æ	Ö	æ	ö				
0	1	1	1	7			§	·	Ç	×	ç	÷				
1	0	0	0	8			"	,	È	Ø	è	ø				
1	0	0	1	9			©	<sup>1</sup>	É	Ù	é	ù				
1	0	1	0	10			ª	º	Ê	Ú	ê	ú				
1	0	1	1	11			«	»	Ë	Û	ë	û				
1	1	0	0	12			¬	<sup>1</sup> / <sub>4</sub>	Ì	Ü	ì	ü				
1	1	0	1	13			SHY	<sup>1</sup> / <sub>2</sub>	Í	Ý	í	ý				
1	1	1	0	14			®	<sup>3</sup> / <sub>4</sub>	Î	Þ	î	þ				
1	1	1	1	15			¯	¿	Ï	ß	ï	ÿ				

Pos.	Name	Note
4/0	CAPITAL LETTER A WITH GRAVE ACCENT	
4/1	CAPITAL LETTER A WITH ACUTE ACCENT	
4/2	CAPITAL LETTER A WITH CIRCUMFLEX ACCENT	
4/3	CAPITAL LETTER A WITH TILDE	
4/4	CAPITAL LETTER A WITH DIAERESIS	
4/5	CAPITAL LETTER A WITH RING ABOVE	
4/6	CAPITAL DIPHTHONG A WITH E	
4/7	CAPITAL LETTER C WITH CEDILLA	
4/8	CAPITAL LETTER E WITH GRAVE ACCENT	
4/9	CAPITAL LETTER E WITH ACUTE ACCENT	
4/10	CAPITAL LETTER E WITH CIRCUMFLEX ACCENT	
4/11	CAPITAL LETTER E WITH DIAERESIS	
4/12	CAPITAL LETTER I WITH GRAVE ACCENT	
4/13	CAPITAL LETTER I WITH ACUTE ACCENT	
4/14	CAPITAL LETTER I WITH CIRCUMFLEX ACCENT	
4/15	CAPITAL LETTER I WITH DIAERESIS	
5/0	CAPITAL LETTER D WITH STROKE	
5/1	CAPITAL LETTER N WITH TILDE	
5/2	CAPITAL LETTER O WITH GRAVE ACCENT	
5/3	CAPITAL LETTER O WITH ACUTE ACCENT	
5/4	CAPITAL LETTER O WITH CIRCUMFLEX ACCENT	
5/5	CAPITAL LETTER O WITH TILDE	
5/6	CAPITAL LETTER O WITH DIAERESIS	
5/7	MULTIPLICATION SIGN	
5/8	CAPITAL LETTER O WITH OBLIQUE STROKE	
5/9	CAPITAL LETTER U WITH GRAVE ACCENT	
5/10	CAPITAL LETTER U WITH ACUTE ACCENT	
5/11	CAPITAL LETTER U WITH CIRCUMFLEX ACCENT	
5/12	CAPITAL LETTER U WITH DIAERESIS	
5/13	CAPITAL LETTER V WITH ACUTE ACCENT	



Pos.	Name	Note
2/0	NO BREAK SPACE	1
2/1	INVERTED EXCLAMATION MARK	
2/2	CENT SIGN	
2/3	POUND SIGN	
2/4	CURRENCY SIGN	
2/5	YEN SIGN	
2/6	BROKEN BAR	
2/7	PARAGRAPH SIGN	
2/8	DIAERESIS	
2/9	COPYRIGHT SIGN	
2/10	FEMININE ORDINAL INDICATOR	
2/11	LEFT ANGLE QUOTATION MARK	
2/12	NOT SIGN	
2/13	SOFT HYPHEN	1
2/14	REGISTERED TRADE MARK SIGN	
2/15	MACRON	
3/0	DEGREE SIGN, RING ABOVE	
3/1	PLUS-MINUS SIGN	
3/2	SUPERSCRIFT TWO	
3/3	SUPERSCRIFT THREE	
3/4	ACUTE ACCENT	
3/5	SMALL GREEK LETTER MU, MICRO SIGN	
3/6	PILCROW SIGN	
3/7	MIDDLE DOT	
3/8	CEDILLA	
3/9	SUPERSCRIFT ONE	
3/10	MASCULINE ORDINAL INDICATOR	
3/11	RIGHT ANGLE QUOTATION MARK	
3/12	VULGAR FRACTION ONE QUARTER	
3/13	VULGAR FRACTION ONE HALF	
3/14	VULGAR FRACTION THREE QUARTERS	
3/15	INVERTED QUESTION MARK	

Pos.	Name	Note
6/0	SMALL LETTER a WITH GRAVE ACCENT	
6/1	SMALL LETTER a WITH ACUTE ACCENT	
6/2	SMALL LETTER a WITH CIRCUMFLEX ACCENT	
6/3	SMALL LETTER a WITH TILDE	
6/4	SMALL LETTER a WITH DIAERESIS	
6/5	SMALL LETTER a WITH RING ABOVE	
6/6	SMALL DIPHTHONG a WITH e	
6/7	SMALL LETTER c WITH CEDILLA	
6/8	SMALL LETTER e WITH GRAVE ACCENT	
6/9	SMALL LETTER e WITH ACUTE ACCENT	
6/10	SMALL LETTER e WITH CIRCUMFLEX ACCENT	
6/11	SMALL LETTER e WITH DIAERESIS	
6/12	SMALL LETTER i WITH GRAVE ACCENT	
6/13	SMALL LETTER i WITH ACUTE ACCENT	
6/14	SMALL LETTER i WITH CIRCUMFLEX ACCENT	
6/15	SMALL LETTER i WITH DIAERESIS	
7/0	SMALL ICELANDIC LETTER ETH	
7/1	SMALL LETTER n WITH TILDE	
7/2	SMALL LETTER o WITH GRAVE ACCENT	
7/3	SMALL LETTER o WITH ACUTE ACCENT	
7/4	SMALL LETTER o WITH CIRCUMFLEX ACCENT	
7/5	SMALL LETTER o WITH TILDE	
7/6	SMALL LETTER o WITH DIAERESIS	
7/7	DIVISION SIGN	
7/8	SMALL LETTER o WITH OBLIQUE STROKE	
7/9	SMALL LETTER u WITH GRAVE ACCENT	
7/10	SMALL LETTER u WITH ACUTE ACCENT	
7/11	SMALL LETTER u WITH CIRCUMFLEX ACCENT	
7/12	SMALL LETTER u WITH DIAERESIS	
7/13	SMALL LETTER y WITH ACUTE ACCENT	
7/14	SMALL ICELANDIC LETTER THORN	
7/15	SMALL LETTER y WITH DIAERESIS	

## NOTES

1

The definition of this character is given in the standards indicated in the ORIGIN field of this registration.



**DEFAULT CGM CHARACTER SET**



## 7.1 Characters of the set and their coded representation

Table 1 — Character set — Coded representation

Bit combi- nation	Name	Bit combi- nation	Name
02/00	SPACE (see 6.3)	06/00	GRAVE ACCENT
02/01	EXCLAMATION MARK	06/01	SMALL LETTER a
02/02	QUOTATION MARK	06/02	SMALL LETTER b
02/03	NUMBER SIGN	06/03	SMALL LETTER c
02/04	DOLLAR SIGN	06/04	SMALL LETTER d
02/05	PERCENT SIGN	06/05	SMALL LETTER e
02/06	AMPERSAND	06/06	SMALL LETTER f
02/07	APOSTROPHE	06/07	SMALL LETTER g
02/08	LEFT PARENTHESIS	06/08	SMALL LETTER h
02/09	RIGHT PARENTHESIS	06/09	SMALL LETTER i
02/10	ASTERISK	06/10	SMALL LETTER j
02/11	PLUS SIGN	06/11	SMALL LETTER k
02/12	COMMA	06/12	SMALL LETTER l
02/13	HYPHEN, MINUS SIGN	06/13	SMALL LETTER m
02/14	FULL STOP	06/14	SMALL LETTER n
02/15	SOLIDUS	06/15	SMALL LETTER o
03/00	DIGIT ZERO	07/00	SMALL LETTER p
03/01	DIGIT ONE	07/01	SMALL LETTER q
03/02	DIGIT TWO	07/02	SMALL LETTER r
03/03	DIGIT THREE	07/03	SMALL LETTER s
03/04	DIGIT FOUR	07/04	SMALL LETTER t
03/05	DIGIT FIVE	07/05	SMALL LETTER u
03/06	DIGIT SIX	07/06	SMALL LETTER v
03/07	DIGIT SEVEN	07/07	SMALL LETTER w
03/08	DIGIT EIGHT	07/08	SMALL LETTER x
03/09	DIGIT NINE	07/09	SMALL LETTER y
03/10	COLON	07/10	SMALL LETTER z
03/11	SEMICOLON	07/11	LEFT CURLY BRACKET
03/12	LESS-THAN SIGN	07/12	VERTICAL LINE
03/13	EQUALS SIGN	07/13	RIGHT CURLY BRACKET
03/14	GREATER-THAN SIGN	07/14	TILDE
03/15	QUESTION MARK	10/00	NO-BREAK SPACE (see 6.3)
04/00	COMMERCIAL AT	10/01	INVERTED EXCLAMATION MARK
04/01	CAPITAL LETTER A	10/02	CENT SIGN
04/02	CAPITAL LETTER B	10/03	POUND SIGN
04/03	CAPITAL LETTER C	10/04	CURRENCY SIGN
04/04	CAPITAL LETTER D	10/05	YEN SIGN
04/05	CAPITAL LETTER E	10/06	BROKEN BAR
04/06	CAPITAL LETTER F	10/07	PARAGRAPH SIGN, SECTION SIGN
04/07	CAPITAL LETTER G	10/08	DIAERESIS
04/08	CAPITAL LETTER H	10/09	COPYRIGHT SIGN
04/09	CAPITAL LETTER I	10/10	FEMININE ORDINAL INDICATOR
04/10	CAPITAL LETTER J	10/11	LEFT ANGLE QUOTATION MARK
04/11	CAPITAL LETTER K	10/12	NOT SIGN
04/12	CAPITAL LETTER L	10/13	SOFT HYPHEN (see 6.3)
04/13	CAPITAL LETTER M	10/14	REGISTERED TRADE MARK SIGN
04/14	CAPITAL LETTER N	10/15	MACRON
04/15	CAPITAL LETTER O	11/00	RING ABOVE, DEGREE SIGN
05/00	CAPITAL LETTER P	11/01	PLUS-MINUS SIGN
05/01	CAPITAL LETTER Q	11/02	SUPERSCRIFT TWO
05/02	CAPITAL LETTER R	11/03	SUPERSCRIFT THREE
05/03	CAPITAL LETTER S	11/04	ACUTE ACCENT
05/04	CAPITAL LETTER T	11/05	MICRO SIGN
05/05	CAPITAL LETTER U	11/06	PILCROW SIGN
05/06	CAPITAL LETTER V	11/07	MIDDLE DOT
05/07	CAPITAL LETTER W	11/08	CEDILLA
05/08	CAPITAL LETTER X	11/09	SUPERSCRIFT ONE
05/09	CAPITAL LETTER Y	11/10	MASCULINE ORDINAL INDICATOR
05/10	CAPITAL LETTER Z	11/11	RIGHT ANGLE QUOTATION MARK
05/11	LEFT SQUARE BRACKET	11/12	VULGAR FRACTION ONE QUARTER
05/12	REVERSE SOLIDUS	11/13	VULGAR FRACTION ONE HALF
05/13	RIGHT SQUARE BRACKET	11/14	VULGAR FRACTION THREE QUARTERS
05/14	CIRCUMFLEX ACCENT	11/15	INVERTED QUESTION MARK
05/15	LOW LINE	12/00	CAPITAL LETTER A WITH GRAVE ACCENT



Table 1 — (concluded)

Bit combination	Name
12/01	CAPITAL LETTER A WITH ACUTE ACCENT
12/02	CAPITAL LETTER A WITH CIRCUMFLEX ACCENT
12/03	CAPITAL LETTER A WITH TILDE
12/04	CAPITAL LETTER A WITH DIAERESIS
12/05	CAPITAL LETTER A WITH RING ABOVE
12/06	CAPITAL DIPHTHONG A WITH E
12/07	CAPITAL LETTER C WITH CEDILLA
12/08	CAPITAL LETTER E WITH GRAVE ACCENT
12/09	CAPITAL LETTER E WITH ACUTE ACCENT
12/10	CAPITAL LETTER E WITH CIRCUMFLEX ACCENT
12/11	CAPITAL LETTER E WITH DIAERESIS
12/12	CAPITAL LETTER I WITH GRAVE ACCENT
12/13	CAPITAL LETTER I WITH ACUTE ACCENT
12/14	CAPITAL LETTER I WITH CIRCUMFLEX ACCENT
12/15	CAPITAL LETTER I WITH DIAERESIS
13/00	CAPITAL ICELANDIC LETTER ETH
13/01	CAPITAL LETTER N WITH TILDE
13/02	CAPITAL LETTER O WITH GRAVE ACCENT
13/03	CAPITAL LETTER O WITH ACUTE ACCENT
13/04	CAPITAL LETTER O WITH CIRCUMFLEX ACCENT
13/05	CAPITAL LETTER O WITH TILDE
13/06	CAPITAL LETTER O WITH DIAERESIS
13/07	MULTIPLICATION SIGN
13/08	CAPITAL LETTER O WITH OBLIQUE STROKE
13/09	CAPITAL LETTER U WITH GRAVE ACCENT
13/10	CAPITAL LETTER U WITH ACUTE ACCENT
13/11	CAPITAL LETTER U WITH CIRCUMFLEX ACCENT
13/12	CAPITAL LETTER U WITH DIAERESIS
13/13	CAPITAL LETTER Y WITH ACUTE ACCENT
13/14	CAPITAL ICELANDIC LETTER THORN
13/15	SMALL GERMAN LETTER SHARP s
14/00	SMALL LETTER a WITH GRAVE ACCENT
14/01	SMALL LETTER a WITH ACUTE ACCENT
14/02	SMALL LETTER a WITH CIRCUMFLEX ACCENT
14/03	SMALL LETTER a WITH TILDE
14/04	SMALL LETTER a WITH DIAERESIS
14/05	SMALL LETTER y WITH RING ABOVE
14/06	SMALL DIPHTHONG e WITH e
14/07	SMALL LETTER c WITH CEDILLA
14/08	SMALL LETTER e WITH GRAVE ACCENT
14/09	SMALL LETTER e WITH ACUTE ACCENT
14/10	SMALL LETTER e WITH CIRCUMFLEX ACCENT
14/11	SMALL LETTER e WITH DIAERESIS
14/12	SMALL LETTER i WITH GRAVE ACCENT
14/13	SMALL LETTER i WITH ACUTE ACCENT
14/14	SMALL LETTER i WITH CIRCUMFLEX ACCENT
14/15	SMALL LETTER i WITH DIAERESIS
15/00	SMALL ICELANDIC LETTER ETH
15/01	SMALL LETTER n WITH TILDE
15/02	SMALL LETTER o WITH GRAVE ACCENT
15/03	SMALL LETTER o WITH ACUTE ACCENT
15/04	SMALL LETTER o WITH CIRCUMFLEX ACCENT
15/05	SMALL LETTER o WITH TILDE
15/06	SMALL LETTER o WITH DIAERESIS
15/07	DIVISION SIGN
15/08	SMALL LETTER o WITH OBLIQUE STROKE
15/09	SMALL LETTER u WITH GRAVE ACCENT
15/10	SMALL LETTER u WITH ACUTE ACCENT
15/11	SMALL LETTER u WITH CIRCUMFLEX ACCENT
15/12	SMALL LETTER u WITH DIAERESIS
15/13	SMALL LETTER y WITH ACUTE ACCENT
15/14	SMALL ICELANDIC LETTER THORN
15/15	SMALL LETTER y WITH DIAERESIS

## 7.2 Code table

The code table (table 2) shows the characters listed at the position in the code table corresponding to the specified bit combination.

The shaded positions correspond to bit combinations that do not represent graphic characters. Their use is outside the scope of ISO 8859; it is specified in other International Standards, for example ISO 646 or ISO 6429.

## 8 Designation of the character set

The graphic characters of this part of ISO 8859 constitute a single coded character set. However, when this character set is implemented together with other coding standards such as ISO 2022 or ISO 4873, the code table of this part of ISO 8859 shall be considered to consist of the following components:

- The character SPACE represented by bit combination 02/00.
- A 94-character G0 graphic character set represented by bit combinations 02/01 to 07/14.
- A 96-character G1 graphic character set represented by bit combinations 10/00 to 15/15.

When required by other coding standards, for example ISO 2022 or ISO 4873, the following pair of escape sequences shall be used:

ESC 02/08 04/02  
ESC 02/13 04/01

to designate the G0 and the G1 sets, respectively. According to ISO 2022, the character SPACE does not require designation.

Table 2 — Code table

b.				0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1			
b.				0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1			
b.				0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1			
b.				0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1			
				00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15			
a. b. b. b.				00	00	00	00	00	SP	0	à	P	`	p			NBSP	°	À	Ð	à	ð
0	0	0	0	01			!	1	A	Q	a	q			i	±	Á	Ñ	á	ñ		
0	0	1	0	02			"	2	B	R	b	r			¢	²	Â	Ò	â	ò		
0	0	1	1	03			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó		
0	1	0	0	04			\$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô		
0	1	0	1	05			%	5	E	U	e	u			¥	µ	Å	Ö	å	ö		
0	1	1	0	06			&	6	F	V	f	v			¦	¶	Æ	Ø	æ	ø		
0	1	1	1	07			'	7	G	W	g	w			§	·	Ç	×	ç	÷		
1	0	0	0	08			(	8	H	X	h	x			¨	,	È	Ø	è	ø		
1	0	0	1	09			)	9	I	Y	i	y			©	¹	É	Ù	é	ù		
1	0	1	0	10			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú		
1	0	1	1	11			+	;	K	Ç	k	ç			«	»	Ë	Û	ë	û		
1	1	0	0	12			,	<	L	\	l	l			¬	¼	Ì	Ü	ì	ü		
1	1	0	1	13			-	=	M	]	m	}			SHY	½	Í	Ý	í	ý		
1	1	1	0	14			.	>	N	~	n	~			®	¾	Î	Þ	î	þ		
1	1	1	1	15			/	?	O	_	o	_			™	¿	Ï	ß	ï	ÿ		





APPLE MACINTOSH

CHARACER SET



Second digit ↓	First digit															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	space	0	@	P	`	p	Ä	ê	†	∞	¿	—		
1	SOH	DC1	!	1	A	Q	a	q	Å	ë	•	±	¡	—		
2	STX	DC2	"	2	B	R	b	r	Ç	í	‡	≤	¬	“		
3	ETX	DC3	#	3	C	S	c	s	É	ì	£	≥	√	”		
4	EOT	DC4	\$	4	D	T	d	t	Ñ	î	§	¥	ƒ	‘		
5	ENQ	NAK	%	5	E	U	e	u	Ö	ï	●	μ	≈	’		
6	ACK	SYN	&	6	F	V	f	v	Ü	ñ	¶	ð	Δ	÷		
7	BEL	ETB	'	7	G	W	g	w	á	ó	ß	Σ	«	◇		
8	BS	CAN	(	8	H	X	h	x	à	ò	®	Π	»	ÿ		
9	HT	EM	)	9	I	Y	i	y	â	ô	©	π	...			
A	LF	SUB	*	:	J	Z	j	z	ä	ö	™	∫	—			
B	VT	ESC	+	;	K	[	k	{	ã	õ	’	ª	À			
C	FF	FS	,	<	L	\			å	ú	™	º	Ã			
D	CR	GS	-	=	M	]	m	}	ç	ù	≠	Ω	Õ			
E	SO	RS	.	>	N	^	n	~	é	û	Æ	æ	Œ			
F	SI	US	/	?	O	_	o	DEL	è	ü	Ø	ø	œ			

— stands for a nonbreaking space, the same width as a digit.

The shaded characters cannot normally be generated from the Macintosh keyboard or keypad.

Figure 1. Macintosh Character Set

Nonprinting or "control" characters (\$00 through \$1F, as well as \$7F) are identified in the table by their traditional ASCII abbreviations; those that are shaded have no special meaning on the Macintosh and cannot normally be generated from the Macintosh keyboard or keypad. Those that can be generated are listed below along with the method of generating them:





**IBM CHARACTER SET**





Character Set (00-7F) Quick Reference

DECIMAL VALUE	HEXA-DECIMAL VALUE	0	16	32	48	64	80	96	112
➡	➡	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	▶	BLANK (SPACE)	0	@	P	,	p
1	1	😊	◀	!	1	A	Q	a	q
2	2	😃	↕	"	2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	♦	¶	\$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	■	&	6	F	V	f	v
7	7	•	↑↓	'	7	G	W	w	w
8	8	☉	↑	(	8	H	X	h	x
9	9	◯	↓	)	9	I	Y	i	y
10	A	⊙	→	*	:	J	Z	j	x
11	B	♂	←	+	;	K	[	k	{
12	C	♀	└	,	<	L	/	l	;
13	D	♪	↔	—	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	♂	▼	/	? :	O	_	o	Δ

Character Set (80-FF) Quick Reference

DECIMAL VALUE	HEXA-DECIMAL VALUE	128	144	160	176	192	208	224	240
➡	➡	8	9	A	B	D	C	E	F
0	0	℥	Ē	ā	☐ 0x00	☐	☐	∞	≡
1	1	ü	Æ	í	☐ 0x01	☐	☐	β	±
2	2	é	FE	ó	☐ 0x02	☐	☐	γ	≥
3	3	â	ô	û	☐	☐	☐	π	≤
4	4	ä	ö	ñ	☐	☐	☐	Σ	∫
5	5	ã	õ	Ñ	☐	☐	☐	σ	∫
6	6	â	û	ä	☐	☐	☐	μ	÷
7	7	ç	ù	ó	☐	☐	☐	τ	≈
8	8	ê	ÿ	ì	☐	☐	☐	Φ	°
9	9	ë	Ö	┐	☐	☐	☐	Θ	•
10	A	è	Ü	┐	☐	☐	☐	Ω	•
11	B	ï	ϕ	½	☐	☐	☐	δ	√
12	C	î	£	¼	☐	☐	☐	∞	η
13	D	ï	¥	í	☐	☐	☐	∅	²
14	E	Ä	Pts	«	☐	☐	☐	∈	■
15	F	Å	f	»	☐	☐	☐	∪	BLANK 'FF'



**IGES 4.0**  
**FONTS AND CHARACTER SETS**





#### 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

**4.2.9 General Note Entity (Type 212).** A General Note Entity consists of one or more text strings. Each text string contains text, a starting point, a text size, and an angle of rotation of the text. Examples of general notes are shown in Figure 64. The FC value indicates the font number and is an integer. Positive values are pre-defined fonts. Negative values point to user defined fonts or modifications to a pre-defined font.

The following fonts are defined:

Font	Description
0	Symbol Font (no longer recommended)
1	Standard Block
2	LeRoy
3	Futura
6	Comp 80
12	News Gothic
13	Lightline Gothic
14	Simplex Roman
17	Century Schoolbook
18	Helvetica
19	OCR-B [ISO1073] (see Appendix J)
1001	Symbol Font 1
1002	Symbol Font 2
1003	Drafting Font (see Appendix J)

Font 0 is an old symbol font and should no longer be used. Figure I1 in Appendix I is a mapping symbol definition for Font 0.

Font 1 does not have a defined display. Use of Font 1 implies the receiving system may use any font which displays the appropriate ASCII format characters. The intent of this font is for usage when the actual display of the characters is not critical for the application.

Font 19 is shown in Figure 65 and is defined in Appendix J (see Section J.5).

Fonts in the 1000 series display symbols mapped onto ASCII characters as shown in Figures 66, 67 and 68. They do not specify a character display font. Font 1003 is defined in Appendix J (see Section J.5).

Table 6 provides names for the graphical characters generated for each valid code.

If the FC number is not sufficient to describe the font, a text font definition entity may be used to define the font. If a text font definition is being used, the negative of the pointer value for the directory entry of the text font definition entity is placed in the FC parameter. The use of the values WT, HT, SL, A, and text start point are shown in Figure 69.

Within definition space, the parameters for the text block are applied in the following order (See Figure 70).

1. Define the box height (HT) and box width (WT).

The rotate internal text flag indicates whether the text box is filled with horizontal text or vertical text. The box width is measured from the start of the left-most (first) text character/symbol in the positive XT direction along the text base line, and extends to the end of the

#### 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

right-most (last) character/symbol, extending  $N$  characters/symbols and  $N-1$  intercharacter spaces. The box height is measured in the positive  $Y^T$  direction and is the height of capital letters. It is equivalent to the symbol "h" used in Appendix C of [ANSI82]. Special symbols, such as those appearing in Appendix C of [ANSI82], which exceed "h" in height are centered vertically. Descenders and portions of symbols exceeding "h" extend outside the lower and/or upper borders of the box. The box height and width are measured before the rotation angle ( $\Lambda$ ) is applied. The text start point is defined as the lower left corner of the first character/symbol box.

When a receiving system cannot fit a text string inside its defined text box, it shall give precedence to maintaining the full original character height, as defined by the text box height.

2. The slant angle is then applied to each individual character. For horizontal text, it is measured from the  $X^T$  axis in a counterclockwise direction. For vertical text, the slant angle is measured from the  $Y^T$  axis.
3. The rotation angle is then applied to the text block. This rotation is applied in a counterclockwise direction about the text start point. The plane of rotation is the  $X^T$ ,  $Y^T$  plane at the depth  $ZS_n$  (where  $ZS_n$  is the value given for the text start point).
4. The mirror operation is performed next. The value 1 indicates the mirror axis is the (rotated) line perpendicular to the text base line and through the text start point. The value 2 indicates the mirror axis is the (rotated) text base line.

Finally, the Transformation Matrix entity is used to specify the relative position of definition space within model space.

The number of characters ( $NC_n$ ) must always be equal to the character count in its corresponding text string ( $TEXT_n$ ).

#### 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

A SINGLE LINE OF TEXT

A GENERAL NOTE  
WITH TWO LINES

MIRRORED TEXT

GENERAL NOTE ROTATED 195 DEGREES

TEXT ROTATED  
195 DEGREES

Figure 64. Examples of the General Note Entity



#### **4.2.9 GENERAL NOTE ENTITY (TYPE 212)**

The definition of this font can be found  
in Appendix J (see Section J.5).

Figure 65. General Note Font 19

#### 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

BL		0	0	e	@	P	P	'	`	p	Ⓟ
!	!	1	1	A	A	q	Q	a	∠	q	¢
"	"	2	2	B	B	R	R	b	⊘	r	⊙
#	#	3	3	C	C	S	S	c	▱	s	Ⓢ
\$	\$	4	4	D	D	T	T	d	◐	t	□
%	%	5	5	E	E	U	U	e	○	u	Ⓢ
&	&	6	6	F	F	V	V	f	//	v	△
'	'	7	7	G	G	W	W	g	⊗	w	◇
(	(	8	8	H	H	X	X	h	↗	x	♠
)	)	9	9	I	I	Y	Y	i	≡	y	⊗
*	*	:	:	J	J	Z	Z	j	⊕	z	Y
+	+	:	:	K	K	[	[	k	⌒	{	{
,	,	<	<	L	L	\	\	l	⊥		
-	-	=	=	M	M	]	]	m	Ⓜ	}	}
.	.	>	>	N	N	-	^	n	∅	-	-
/	/	?	?	O	O	-	-	o	○		

Figure 66. General Note Font 1001

#### 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

BL		0	0	e	@	P	P	'	`	p	†
!	!	1	1	A	A	q	Q	a	∞	q	‡
"	"	2	2	B	B	R	R	b	÷	r	→
#	±	3	3	C	C	S	S	c	≤	s	←
\$	•	4	4	D	D	T	T	d	≥	t	φ
%	%	5	5	E	E	U	U	e	Δ	u	θ
&	&	6	6	F	F	V	V	f	√	v	γ
,	,	7	7	G	G	W	W	g	×	w	ψ
(	(	8	8	H	H	X	X	h	≡	x	ω
)	)	9	9	I	I	Y	Y	i	≠	y	λ
*	*	:	:	J	J	Z	Z	j	∫	z	α
+	+	;	;	K	K	[	[	k	⊃	{	δ
,	,	<	<	L	L	\	\	l	∇	l	μ
-	-	=	=	M	M	]	]	m	∧	}	π
.	.	>	>	N	N	^	^	n	≈	-	—
/	/	?	?	O	O	-	-	o	Σ		

Figure 67. General Note Font. 1002

#### 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

The definition of this font can be found  
in Appendix J (see Section J.5).

Figure 68. General Note Font 1003



#### 4.2.9 GENERAL NOTE ENTITY (TYPE 212)









Table 6. Character Names for the Symbol and Drafting Fonts

Name	Symbol	Font†			
		1	1001	1002	1003
Space		32	32	32	see
Exclamation mark	!	33	33	33	App. J
Quotation marks	"	34	34	34	for this
Pound sign	#	35	35		Font
Plus/minus	±			35	
Dollar sign	\$	36	36		
Degree symbol	°			36	
Percent sign	%	37	37	37	
Ampersand	&	38	38	38	
Apostrophe	'	39	39	39	
Left parenthesis	(	40	40	40	
Right parenthesis	)	41	41	41	
Asterisk	*	42	42	42	
Plus sign	+	43	43	43	
Comma	,	44	44	44	
Minus sign/hyphen	-	45	45	45	
Period	.	46	46	46	
Slash	/	47	47	47	
Numeric 0	0	48	48	48	
Numeric 1	1	49	49	49	
Numeric 2	2	50	50	50	
Numeric 3	3	51	51	51	
Numeric 4	4	52	52	52	
Numeric 5	5	53	53	53	
Numeric 6	6	54	54	54	
Numeric 7	7	55	55	55	
Numeric 8	8	56	56	56	
Numeric 9	9	57	57	57	
Colon	:	58	58	58	
Semi-colon	;	59	59	59	
Less than	<	60	60	60	
Equal sign	=	61	61	61	
Greater than	>	62	62	62	
Question mark	?	63	63	63	
Commercial at	@	64	64	64	
Upper case letter A	A	65	65	65	
Upper case letter B	B	66	66	66	
Upper case letter C	C	67	67	67	
Upper case letter D	D	68	68	68	
Upper case letter E	E	69	69	69	
Upper case letter F	F	70	70	70	
Upper case letter G	G	71	71	71	
Upper case letter H	H	72	72	72	

†Entries for each Font are decimal ASCII equivalent

## 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

Table 6. Character Names for the Symbol and Drafting Fonts (continued)

Name	Symbol	Font†			
		1	1001	1002	1003
Upper case letter I	I	73	73	73	see App. J for this Font
Upper case letter J	J	74	74	74	
Upper case letter K	K	75	75	75	
Upper case letter L	L	76	76	76	
Upper case letter M	M	77	77	77	
Upper case letter N	N	78	78	78	
Upper case letter O	O	79	79	79	
Upper case letter P	P	80	80	80	
Upper case letter Q	Q	81	81	81	
Upper case letter R	R	82	82	82	
Upper case letter S	S	83	83	83	
Upper case letter T	T	84	84	84	
Upper case letter U	U	85	85	85	
Upper case letter V	V	86	86	86	
Upper case letter W	W	87	87	87	
Upper case letter X	X	88	88	88	
Upper case letter Y	Y	89	89	89	
Upper case letter Z	Z	90	90	90	
Left bracket	[	91	91	91	
Backward slash	\	92	92	92	
Right bracket	]	93	93	93	
Caret	^	94	94	94	
Underscore	_	95	95	95	
Reverse quote	`	96	96	96	
Lower case letter a	a	97			
Angularity			97		
Marker/symbol				97	
Lower case letter b	b	98			
Marker/symbol			98		
Division symbol	÷			98	
Lower case letter c	c	99			
Flatness			99		
Less than or equal	≤			99	
Lower case letter d	d	100			
Profile of a surface			100		
Greater than or equal	≥			100	
Lower case letter e	e	101			
Circularity			101		
Marker/symbol				101	
Lower case letter f	f	102			
Parallelism			102		
Radical	√			102	

†Entries for each Font are decimal ASCII equivalent

## 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

Table 6. Character Names for the Symbol and Drafting Fonts (continued)

Name	Symbol	Font†			
		1	1001	1002	1003
Lower case letter g	g	103			see
Cylindricity			103		App. J for this Font
Cross product	x			103	
Lower case letter h	h	104			
Runout			104		
Congruence	≡			104	
Lower case letter i	i	105			
Symmetry	≡		105		
Not equal	≠			105	
Lower case letter j	j	106			
Position	⊕		106		
Integral	∫			106	
Lower case letter k	k	107			
Profile of a line	(		107		
Implication	⊃			107	
Lower case letter l	l	108			
Perpendicularity	⊥		108		
Union	∪			108	
Lower case letter m	m	109			
Maximum material condition	Ⓜ		109		
Intersection	∧			109	
Lower case letter n	n	110			
Diameter	∅		110		
Approximately equal	≈			110	
Lower case letter o	o	111			
All around applicability	⊙		111		
Greek letter sigma (Sum)	Σ			111	
Lower case letter p	p	112			
Projected tolerance zone	Ⓟ		112		
Up arrow	↑			112	
Lower case letter q	q	113			
Centerline	⌒		113		
Down arrow	↓			113	
Lower case letter r	r	114			
Concentricity	⊙		114		
Right arrow	→			114	
Lower case letter s	s	115			
Regardless of feature side	Ⓢ		115		
Left arrow	←			115	
Lower case letter t	t	116			
Marker/symbol	□		116		
Greek letter phi	φ			116	

† Entries for Each Font are decimal ASCII equivalent

#### 4.2.9 GENERAL NOTE ENTITY (TYPE 212)

Table 6. Character Names for the Symbol and Drafting Fonts (continued)

Name	Symbol	Font†			
		1	1001	1002	1003
Lower case letter u	u	117			see App. J for this Font
Marker/symbol	⊙		117		
Greek letter theta	θ			117	
Lower case letter v	v	118			
Marker/symbol	△		118		
Greek letter gamma	γ			118	
Lower case letter w	w	119			
Marker/symbol	◇		119		
Greek letter psi	ψ			119	
Lower case letter x	x	120			
Marker/symbol	⊕		120		
Greek letter omega	ω			120	
Lower case letter y	y	121			
Marker/symbol	⊗		121		
Greek letter lambda	λ			121	
Lower case letter z	z	122			
Marker/symbol	Y		122		
Greek letter alpha	α			122	
Left brace	{	123	123		
Greek letter delta	δ			123	
Vertical bar		124	124		
Greek letter mu	μ			124	
Right brace	}	125	125		
Greek letter pi	π			125	
Tilde	-	126	126		
Overscore	—			126	

†Entries for each Font are decimal ASCII equivalent



## APPENDIX J.5 ADDITIONAL GENERAL NOTE FONTS

### J.5 Additional General Note Fonts

Two additional fonts—Font 19 (OCR-B) and Font 1003 (Drafting Font)—are defined for use by the General Note Entity (Type 212). See Section 4.2.9 for more detail.

Font 19 (OCR-B) [ISO1073] is defined in Figure J1.

Font 1003 (Drafting Font) is defined in Figure J2 and Table J2.

BL		o	O	e	@	P	P	'	`	p	p
!	!	1	1	A	A	q	Q	a	a	q	q
"	"	2	2	B	B	R	R	b	b	r	r
#	#	3	3	C	C	S	S	c	c	s	s
\$	\$	4	4	D	D	T	T	d	d	t	t
%	%	5	5	E	E	U	U	e	e	u	u
&	&	6	6	F	F	V	V	f	f	v	v
'	'	7	7	G	G	W	W	g	g	w	w
(	(	8	8	H	H	X	X	h	h	x	x
)	)	9	9	I	I	Y	Y	i	i	y	y
*	*	:	:	J	J	Z	Z	j	j	z	z
+	+	:	:	K	K	[	[	k	k	{	{
,	,	<	<	L	L	\	\	l	l		
-	-	=	=	M	M	]	]	m	m	}	}
.	.	>	>	N	N	^	^	n	n	~	~
/	/	?	?	O	O	-	-	o	o		

Figure J1. General Note Font 19 (OCR-B)

# APPENDIX J.5 ADDITIONAL GENERAL NOTE FONTS

BL		0	0	o	Q	P	P	'	±	p	Ⓟ
!	!	1	1	A	A	q	Q	a	∠	q	¢
"	"	2	2	B	B	R	R	b	⊥	r	⊙
#	#	3	3	C	C	S	S	c	▱	s	Ⓢ
\$	\$	4	4	D	D	T	T	d	⌒	t	⚡
%	%	5	5	E	E	U	U	e	○	u	—
&	&	6	6	F	F	V	V	f	//	v	└
,	,	7	7	G	G	W	W	g	↗	w	∇
(	(	8	8	H	H	X	X	h	↖	x	⊥
)	)	9	9	I	I	Y	Y	i	≡	y	⇒
*	*	:	:	J	J	Z	Z	j	⊕	z	△
+	+	;	;	K	K	[	[	k	⌒	{	{
,	,	<	<	L	L	\	\	l	Ⓛ		
-	-	=	=	H	M	]	]	m	Ⓜ	}	}
.	.	>	>	N	N	-	-	n	∅	-	•
/	/	?	?	O	O	-	-	o	□		

Figure J2. General Note Font 1003 (Drafting Font)

# APPENDIX J.5 ADDITIONAL GENERAL NOTE FONTS

Table J2. Character Names for the Drafting Font

Name	Symbol	Font†
		1003
Space		32
Exclamation mark	!	33
Quotation marks	"	34
Pound sign	#	35
Dollar sign	\$	36
Percent sign	%	37
Ampersand	&	38
Apostrophe	'	39
Left parenthesis	(	40
Right parenthesis	)	41
Asterisk	*	42
Plus sign	+	43
Comma	,	44
Minus sign/hyphen	-	45
Period	.	46
Slash	/	47
Numeric 0	0	48
Numeric 1	1	49
Numeric 2	2	50
Numeric 3	3	51
Numeric 4	4	52
Numeric 5	5	53
Numeric 6	6	54
Numeric 7	7	55
Numeric 8	8	56
Numeric 9	9	57
Colon	:	58
Semi-colon	;	59
Less than	<	60
Equal sign	=	61
Greater than	>	62
Question mark	?	63
Commercial at	@	64
Upper case letter A	A	65
Upper case letter B	B	66
Upper case letter C	C	67
Upper case letter D	D	68
Upper case letter E	E	69
Upper case letter F	F	70
Upper case letter G	G	71
Upper case letter H	H	72
Upper case letter I	I	73
Upper case letter J	J	74

†Entries are decimal ASCII equivalent

## APPENDIX J.5 ADDITIONAL GENERAL NOTE FONTS

Table J2. Character Names for the Drafting Font (continued)












Name	Symbol	Font†
		1003
Upper case letter K	K	75
Upper case letter L	L	76
Upper case letter M	M	77
Upper case letter N	N	78
Upper case letter O	O	79
Upper case letter P	P	80
Upper case letter Q	Q	81
Upper case letter R	R	82
Upper case letter S	S	83
Upper case letter T	T	84
Upper case letter U	U	85
Upper case letter V	V	86
Upper case letter W	W	87
Upper case letter X	X	88
Upper case letter Y	Y	89
Upper case letter Z	Z	90
Left bracket	[	91
Backward slash	\	92
Right bracket	]	93
Arc length	(	94
Underscore	_	95
Plus/minus	±	96
Angularity	∠	97
Perpendicularity	⊥	98
Flatness	▯	99
Profile of a surface	⌀	100
Circularity	○	101
Parallelism	∥	102
Cylindricity	⌀	103
Runout	↗	104
Symmetry	≡	105
Position	⊕	106
Profile of a line	⌒	107
Least material condition	Ⓛ	108
Maximum material condition	Ⓜ	109
Diameter	⌀	110
Square	□	111
Projected tolerance zone	Ⓟ	112
Centerline	⌵	113
Concentricity	◎	114
Regardless of feature side	Ⓢ	115

†Entries are decimal ASCII equivalent



## APPENDIX J.5 ADDITIONAL GENERAL NOTE FONTS

Table J2. Character Names for the Drafting Font (continued)

Name	Symbol	Font†
		1003
Total runout		116
Straightness		117
Counterbore		118
Countersink		119
Depth		120
Conical taper		121
Slope		122
Left brace		123
Vertical bar		124
Right brace		125
Degree symbol		126

†Entries are decimal ASCII equivalent

**APPENDIX B**  
**FONT EXAMPLES**



**ACCENTED LATIN CHARACTERS**





### A.3.2 accented Latin characters (ID's = 61696 +)

These character identifiers are reserved for accented Latin letters.

Note: For ease of reading, only the low order 8 binary bits of the character identifier number is notated. Therefore, please add the following to each number shown:  
Octal: 361000; Decimal: 61696; Hex: F100.

IDENTIFIER			SHAPE	CHARACTER DESCRIPTION
Octal	Dec	Hex		
41 <sub>8</sub>	33	21	À	Grave A
42 <sub>8</sub>	34	22	Á	Acute A
43 <sub>8</sub>	35	23	Â	Circumflex A
44 <sub>8</sub>	36	24	Ã	Tilde A
45 <sub>8</sub>	37	25	Ä	Macron A
46 <sub>8</sub>	38	26	Å	Breve A
47 <sub>8</sub>	39	27	Ä	Diaeresis A = umlaut A
50 <sub>8</sub>	40	28	Å	Ring A = angstrom A <AMS, AA>
51 <sub>8</sub>	41	29	Ą	Ogonek A
52 <sub>8</sub>	42	2A	Ĉ	Acute C
53 <sub>8</sub>	43	2B	Ĉ	Circumflex C
54 <sub>8</sub>	44	2C	Ĉ	High dot C
55 <sub>8</sub>	45	2D	Ç	Cedilla C
56 <sub>8</sub>	46	2E	Č	Hachek C = caron C
57 <sub>8</sub>	47	2F	Ď	Hachek D = caron D
60 <sub>8</sub>	48	30	È	Grave E
61 <sub>8</sub>	49	31	É	Acute E
62 <sub>8</sub>	50	32	Ê	Circumflex E
63 <sub>8</sub>	51	33	Ë	Macron E
64 <sub>8</sub>	52	34	Ë	High dot E
65 <sub>8</sub>	53	35	Ë	Diaeresis E = umlaut E
66 <sub>8</sub>	54	36	Ę	Ogonek E
67 <sub>8</sub>	55	37	Ě	Hachek E = caron E
70 <sub>8</sub>	56	38	Ĝ	Acute G
71 <sub>8</sub>	57	39	Ĝ	Circumflex G
72 <sub>8</sub>	58	3A	Ğ	Breve G
73 <sub>8</sub>	59	3B	Ĝ	High dot G
74 <sub>8</sub>	60	3C	Ġ	Cedilla G
75 <sub>8</sub>	61	3D	Ĥ	Circumflex H
76 <sub>8</sub>	62	3E	Ì	Grave I
77 <sub>8</sub>	63	3F	Í	Acute I

IDENTIFIER			SHAPE	CHARACTER DESCRIPTION
Octal	Dec	Hex.		
100 <sub>8</sub>	64	40	Î	Circumflex I
101 <sub>8</sub>	65	41	Ĩ	Tilde I
102 <sub>8</sub>	66	42	Ī	Macron I
103 <sub>8</sub>	67	43	İ	Ihigh dot I
104 <sub>8</sub>	68	44	Ï	Diaeresis I = umlaut I
105 <sub>8</sub>	69	45	Į	Ogonek I
106 <sub>8</sub>	70	46	Ĵ	Circumflex J
107 <sub>8</sub>	71	47	Ķ	Cedilla K
110 <sub>8</sub>	72	48	Ĺ	Acute L
111 <sub>8</sub>	73	49	Ľ	Cedilla L
112 <sub>8</sub>	74	4A	Ľ̃	Hachek L = caron L
113 <sub>8</sub>	75	4B	Ň	Acute N
114 <sub>8</sub>	76	4C	Ñ	Tilde N
115 <sub>8</sub>	77	4D	Ŋ	Cedilla N
116 <sub>8</sub>	78	4E	Ñ̃	Hachek N = caron N
117 <sub>8</sub>	79	4F	Ò	Grave O
120 <sub>8</sub>	80	50	Ó	Acute O
121 <sub>8</sub>	81	51	Ô	Circumflex O
122 <sub>8</sub>	82	52	Õ	Tilde O
123 <sub>8</sub>	83	53	Ō	Macron O
124 <sub>8</sub>	84	54	Ö	Diaeresis O = umlaut O
125 <sub>8</sub>	85	55	Ȫ	Double acute O
126 <sub>8</sub>	86	56	Ŕ	Acute R
127 <sub>8</sub>	87	57	Ŗ	Cedilla R
130 <sub>8</sub>	88	58	Ŗ̃	Hachek R = caron R
131 <sub>8</sub>	89	59	Ŝ	Acute S
132 <sub>8</sub>	90	5A	Š	Circumflex S
133 <sub>8</sub>	91	5B	Ṣ̌	Cedilla S
134 <sub>8</sub>	92	5C	Š̃	Hachek S = caron S
135 <sub>8</sub>	93	5D	Ţ	Cedilla T
136 <sub>8</sub>	94	5E	Ţ̃	Hachek T = caron T
137 <sub>8</sub>	95	5F	Ù	Grave U
140 <sub>8</sub>	96	60	Ú	Acute U
141 <sub>8</sub>	97	61	Û	Circumflex U
142 <sub>8</sub>	98	62	Ũ	Tilde U
143 <sub>8</sub>	99	63	Ū	Macron U
144 <sub>8</sub>	100	64	Ů	Breve U
145 <sub>8</sub>	101	65	Ü	Diaeresis U = umlaut U

ASSOCIATION FOR FONT  
INFORMATION EXCHANGE

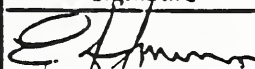






Association for Font Information Interchange

For AFII Use Only

GlyphID (Decimal)				New	Old	Rev.	Den.	Date	Signature
000	000	253	251	X				7/15/88	

Action Taken: New glyph assignment. Definition—"French Franc = Alternate rendition of 'Francs' (357<sub>8</sub>|243<sub>8</sub>)".

## Application for Assignment of a GlyphID

Please answer the following questions as best as possible. The more information provided the more likely that a correct registration of the glyph(s) can be made. Place hand drawn and printed samples of the glyph on the last two pages of this registration.

### 1. Registrant

Name: Ronald J. Pellar (representative to ISO JTC1/SC2/WG2)

Organization: Xerox Corporation ESAE 322

Address: 701 S. Aviation Blvd.

City: El Segundo

State, District, or Province and Postal Code: California 90245

Country: USA

Telephone, Telex, and/or FAX #: (213) 333-7364

### 2. Glyph name in native language. Franc

---

---

---

---

### 3. Glyph name transliterated or transcribed into the latin alphabet. Franc

---

---

*Association for Font Information Interchange*

- \_\_\_\_\_
- \_\_\_\_\_
4. What script, alphabet, syllibary, if any, is the glyph used with? Latin Alphabet
- \_\_\_\_\_
- \_\_\_\_\_
5. What language, if any, is the glyph used with? French
- \_\_\_\_\_
- \_\_\_\_\_
6. What category do the users of this glyph belong? (Please circle or underline choice(s))  
Mathematics      Linguistic      Technical      Other Monetary
7. This glyph is similar in shape to what other glyph already in the registry?  
(GlyphID, if possible) The Latin Letter F (000|000|000|070)
8. This glyph is similar in meaning to what other glyph already in the registry?  
(GlyphID, if possible) Francs (000|000|239|163)
9. Are there unique metrics associated with this glyph? (YES or NO) Yes/No
10. Metrics: (Please denote measuring scheme and units) Font dependant
- a. Side bearings N/A
- b. Width Maybe on a figure width
- c. Height Numeral Height
- d. Design size N/A
11. Does this glyph vary in any of the following ways, if so how? Yes
- a. Weight Font dependant
- b. Italic version Font dependant
- c. Slanting Font dependant
- d. Scalability Scalable like rest of numerals
12. Is this glyph design dependant or design transparent, i.e., does the shape change with the design? Design Dependant

*Association for Font Information Interchange*

13. Are there size limitations on this glyph? (YES or NO) No

a. If so what are they \_\_\_\_\_

14. Does this glyph exist in electronic format, either bit map or contour? (YES or NO) Yes

If so where, what format, and may it be submitted with release for use by AFII?

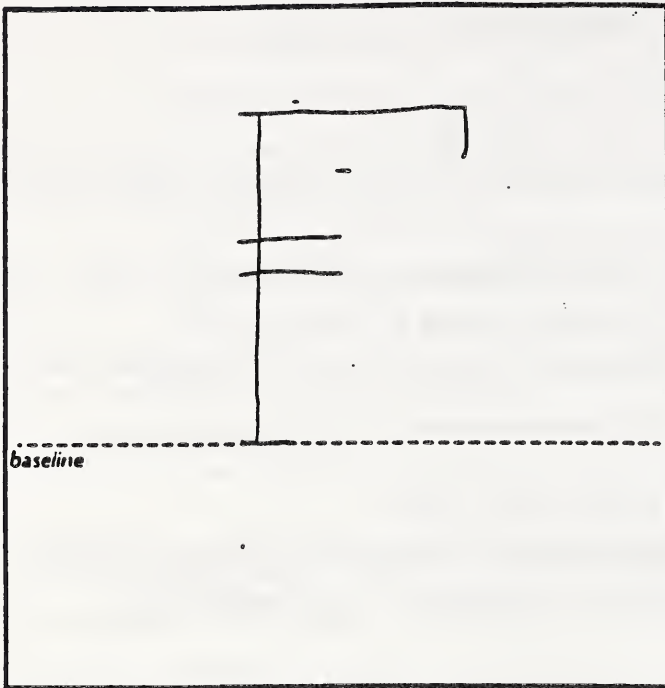
Unknown

15. Additional comment(s), or information relating to this glyph:

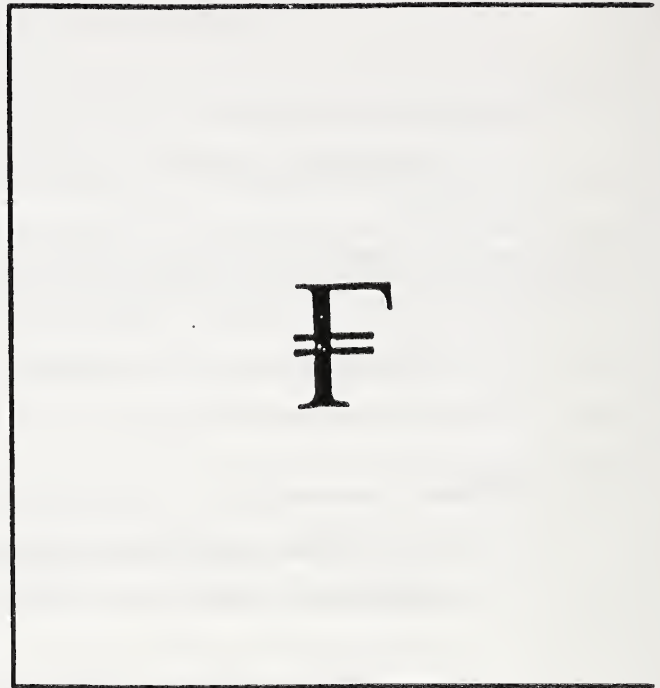
This glyph has been designated as the official monetary symbol  
for the French Franc. It is used on the new One Franc coin.



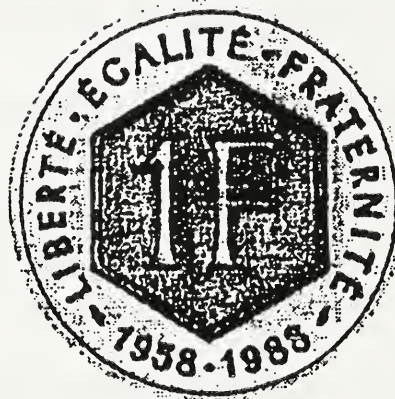
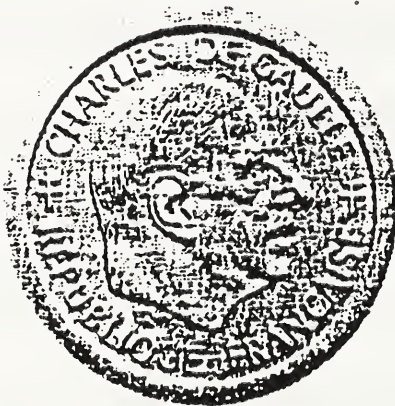
*Association for Font Information Interchange*



Hand drawn sample



Large printed sample

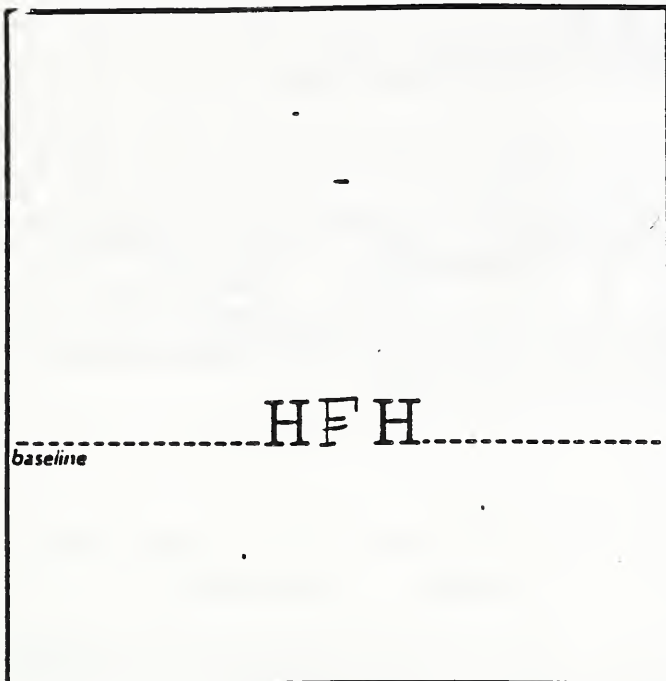


£ \$ ¢ ¥ ₣

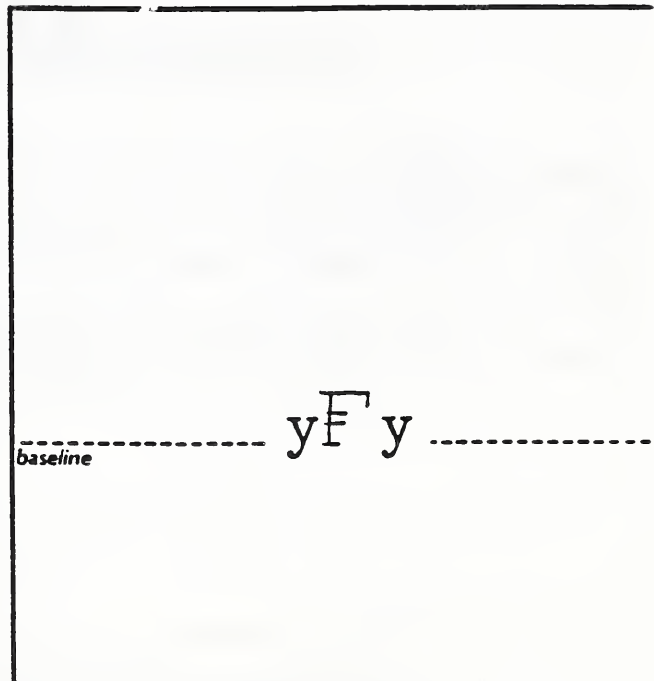
The new French franc, honoring De Gaulle and introducing the new franc symbol.

Typical printed sample showing the Glyph in context, or typical use

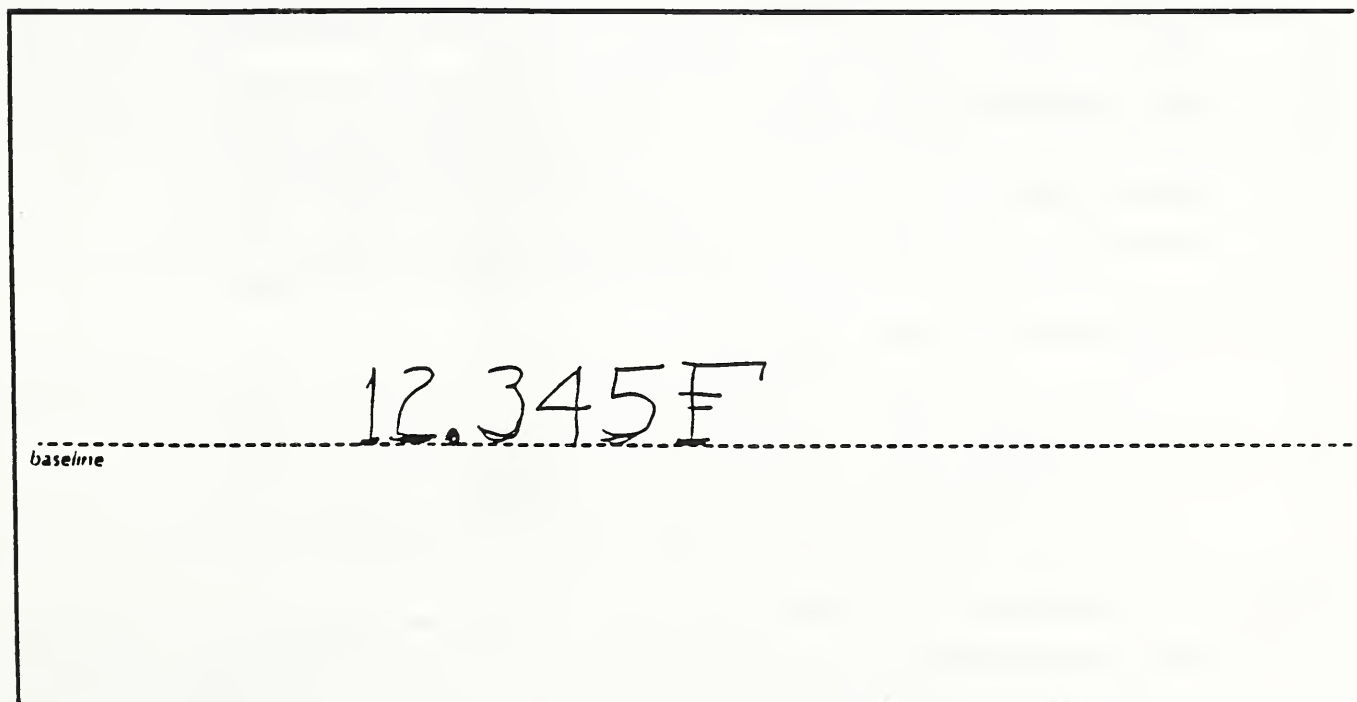
## Association for Font Information Interchange



- Hand drawn sample between H's.  
Try to show relative height,  
width, and placement of glyph.  
Add notes or comments,  
if necessary.



- Hand drawn sample between y's.  
Try to show relative height,  
width, and placement of glyph.  
Add notes or comments,  
if necessary.



Handdrawn sample of how glyph might be used  
in a typical context.



Association for Font Information Interchange

*For AFII Use Only*

GlyphID (Decimal)				New	Old	Rev.	Den.	Date	Signature
000	000	035	099	X				7/15/88	<i>[Signature]</i>

Action Taken: New glyph assignment. Definition-"Hausa small letter K (Small letter velar unvoiced ejective)". Parenthetical definition extracted from ISO 6438.

## Application for Assignment of a GlyphID

Please answer the following questions as best as possible. The more information provided the more likely that a correct registration of the glyph(s) can be made. Place hand drawn and printed samples of the glyph on the last two pages of this registration.

### 1. Registrant

Name: Ronald J. Pellar

Organization: Xerox Corporation

Address: 701 S. Aviation Blvd.

City: El Segundo

State, District, or Province and Postal Code: California 90245

Country: USA

Telephone, Telex, and/or FAX #: (213) 333-7364

### 2. Glyph name in native language. Unknown

---

---

---

---

### 3. Glyph name transliterated or transcribed into the latin alphabet. Unknown

---

---

Association for Font Information Interchange

4. What script, alphabet, syllibary, if any, is the glyph used with? Latin Alphabet

5. What language, if any, is the glyph used with? Hausa

6. What category do the users of this glyph belong? (Please circle or underline choice(s))  
Mathematics      Linguistic      Technical      Other \_\_\_\_\_

7. This glyph is similar in shape to what other glyph already in the registry?  
(GlyphID, if possible) The Latin Letter k (000|000|000|107)

8. This glyph is similar in meaning to what other glyph already in the registry?  
(GlyphID, if possible) The Latin Letter k (000|000|000|107)

9. Are there unique metrics associated with this glyph? (YES or NO) No

10. Metrics: (Please denote measuring scheme and units) Font dependant  
a. Side bearings Same as Latin letter k.  
b. Width Same as Latin letter k.  
c. Height Same as Latin letter k.  
d. Design size Same as Latin letter k.

11. Does this glyph vary in any of the following ways, if so how? Yes  
a. Weight Font dependant. Same as Latin letter k.  
b. Italic version Font dependant. Same as Latin letter k.  
c. Slanting Font dependant. Same as Latin letter k.  
d. Scalability Scalable like rest of alphabet.

12. Is this glyph design dependant or design transparent, i.e., does the shape change with font design? Design Dependant



**Association for Font Information Interchange**

13. Are there size limitations on this glyph? (YES or NO) No

a. If so what are they \_\_\_\_\_

14. Does this glyph exist in electronic format, either bit map or contour? (YES or NO) Yes

If so where, what format, and may it be submitted with release for use by AFII?

This glyph can be obtained in Ikarus format and the letter it  
represents is in the public domain.

15. Additional comment(s), or information relating to this glyph:

---

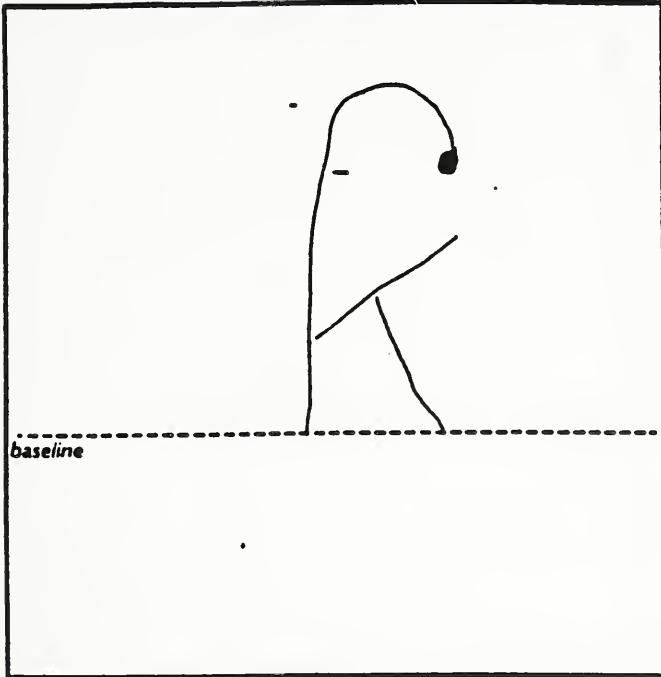
---

---

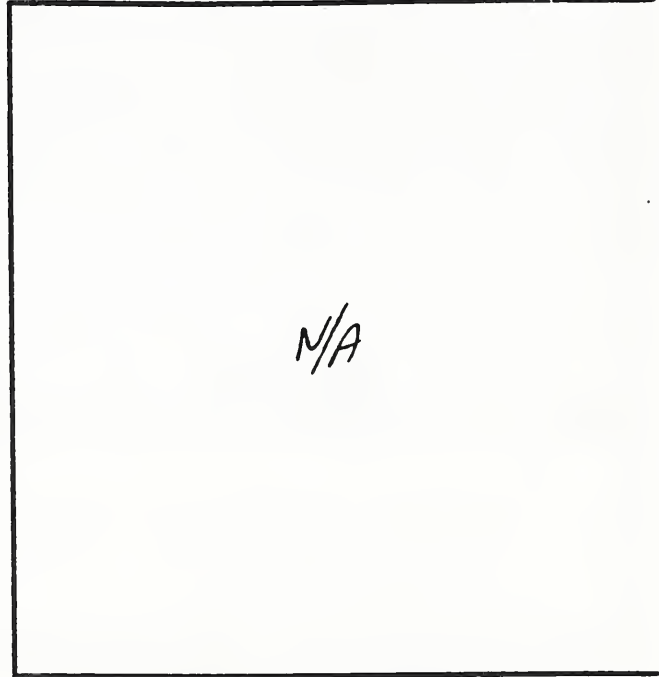
---

---

## Association for Font Information Interchange



Hand drawn sample



Large printed sample

### **HAUSA (Latin script)**

Hausa is the second most widely spoken language in Africa. In Nigeria alone it is probably spoken and understood by ten million people.

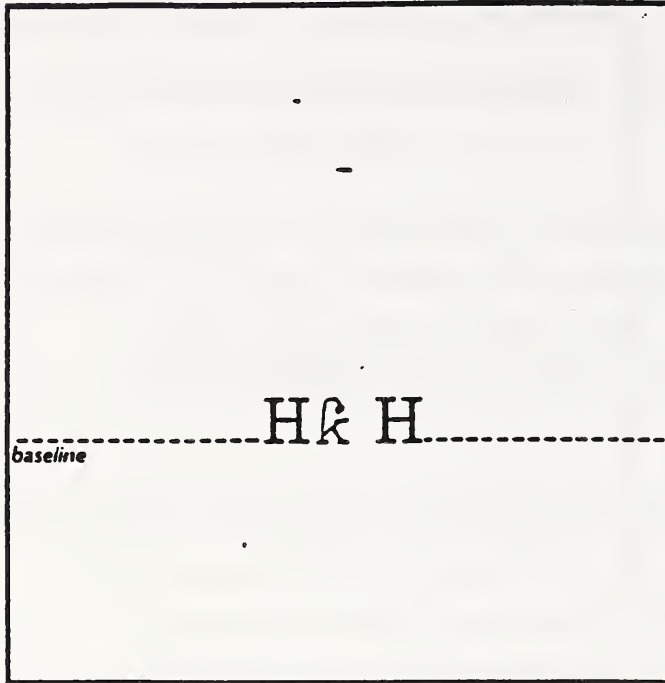
It is written in Latin script, using the following special letters: *ɓ* & *ƙ*.

**SPECIMEN** This is part of a Hausa folk-tale:

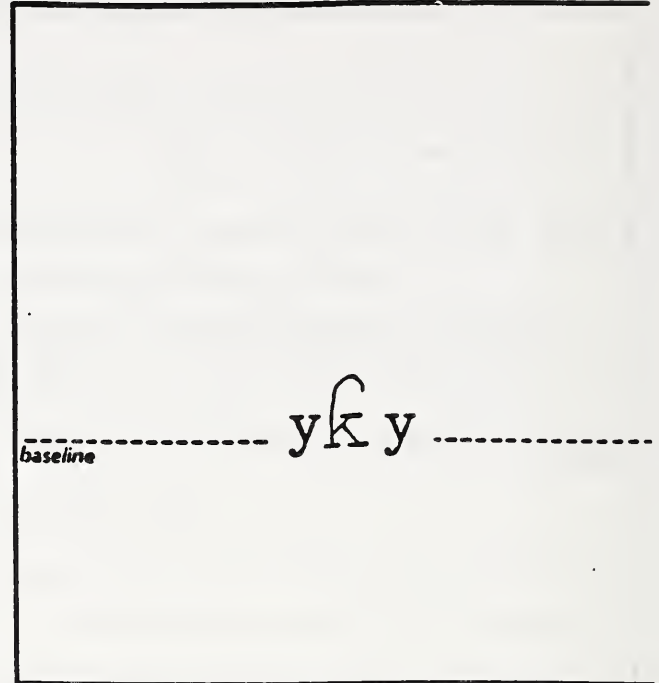
Wannan wani mutum ke nan, raƙumara ta ɓace. Sai ta kama hanya, tana tafiya. Tana tafe tana figar ganyen itace, tana ci.

Typical printed sample showing the Glyph in context, or typical use

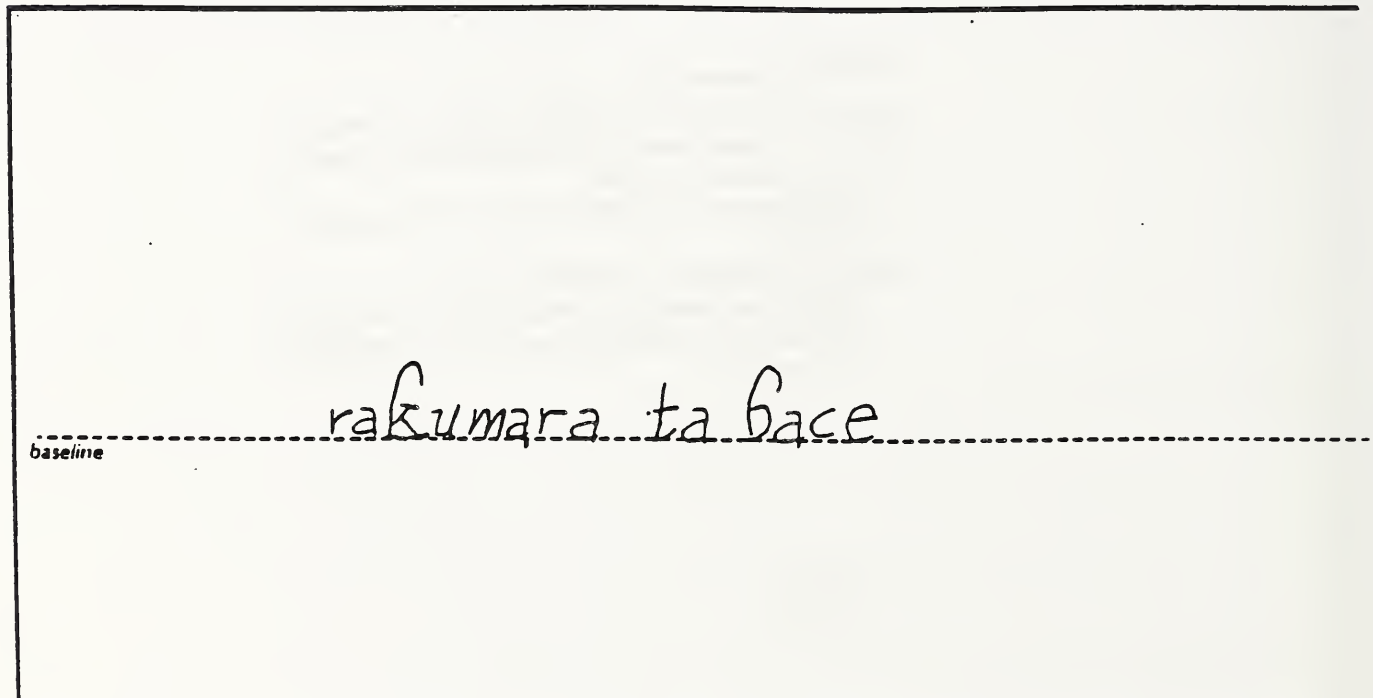
## Association for Font Information Interchange



Hand drawn sample between H's.  
Try to show relative height,  
width, and placement of glyph.  
Add notes or comments,  
if necessary.



Hand drawn sample between y's.  
Try to show relative height,  
width, and placement of glyph.  
Add notes or comments,  
if necessary.



Handdrawn sample of how glyph might be used  
in a typical context.

**EXAMPLES OF TYPOGRAPHIC  
QUALITY FONTS FROM ITC**





10  
 2G  
 4A  
 VNT  
 SST  
 UT  
 W  


---

 W

AAAC  
 EAFARG  
 HIKALALA  
 LMINNT  
 RRASSST  
 STTHUT  
 VVNVW  


---

 cetvvv

nnopq  
JKLM

kmn  
HIJK

hijk  
FGH

ghi  
EFG

efh  
DEI

de  
JD

abcddeefghijklmnopqrstuvwxyz  
ABCDEFGHIJKLMNPOQRSTUVWXYZ  
AABBBCCDEFGHIJKL MNOPQ  
RRSTUVVWVWXYZ  
Tj6fñkyltqlpwy6m  
1234567890&@&1234567890  
(:;!?'^%#&@&)

Alternate characters are shown immediately following regular font characters.  
Lining figures are shown at left; Old Style figures at right.

26  
good  
reasons  
to use  
ITC Bookman  
Outline  
with  
Swashes

**APPLE LASER WRITER FONTS**





I ©  
 G A  
 A L A  
 V N T  
 S S T  
 U T  
 W W  


---

 v v

A A C A C ©  
 E A F A R G A  
 H T K A L A L A  
 L M N N N I  
 R R R A S S T  
 S T T H U T  
 V V W N W  


---

 c e t v w y

---

**System Requirements**

To use the Apple LaserWriter II<sup>mx</sup> printer, you must have one of these systems:

► One or more Macintosh (minimum 512K of RAM) or Apple II<sup>cs</sup> computers connected via the LocalTalk Cabling System.

► An MS-DOS or OS/2 computer with a LocalTalk PC Card or an RS-232-C cable and appropriate software.  
► Any other computer with an RS-232-C cable and appropriate software.

---

**Technical Specifications****Marking engine**

► Canon LBP-SX laser xerographic

**Processor**

► Motorola 68020 (16.67-megahertz clock speed)

**Memory**

► 1 megabyte ROM;  
2 megabytes RAM

**Interfaces**

► SCSI, AppleTalk, Apple Desktop Bus™ (for future expansion), and RS-232-C ports

**Expansion capabilities**

► ROM expansion via font-expansion slot  
► RAM expansion up to 12 megabytes  
► External SCSI port for hard disk font storage  
► Apple Desktop Bus for future expansion

**Print quality**

► All text and graphics printed at 300 by 300 dots per inch, full page

**Built-in font families**

► Times, Helvetica, Courier, Symbol, ITC Avant Garde Gothic, ITC Bookman, New Century Schoolbook, Helvetica Narrow, Palatino, ITC Zapf Chancery, and ITC Zapf Dingbats

**Speed**

► 8 pages per minute maximum throughput (actual speed depends on images printed)

**Printing protocols**

► PostScript, a subset of the Diablo 630 command set, and Hewlett-Packard LaserJet Plus emulation

**Print materials**

► Letter, legal, A4, and B5 sizes, using 16- to 20-pound single-sheet photocopy bond, 8- to 34-pound letterhead and colored stock, or transparency overhead film. Envelopes, labels, and paper (up to 36-pound) supported via manual feed. Envelopes also supported via optional envelope tray.

**Print capacities**

► Paper cassettes hold 200 sheets of 20-pound paper.  
► Envelope cassette holds 15 envelopes.

**Printable surface**

► Letter size: 8.0 by 10.5 inches; legal: 8.0 by 13.0 inches; A4: 7.41 by 10.86 inches; B5: 7.69 by 10.16 inches (actual printable area may vary depending on application)

**Size and weight**

► Height: 8.6 in. (21.8 cm)  
► Width: 20 in. (50.8 cm)  
With letter tray attached, 26.4 in. (67.1 cm)  
► Depth: 18.5 in. (47 cm)  
► Weight: 45 lb. (20.25 kg)

**Operating environment**

► Temperature: 50° to 90° F (10° to 32° C)  
► Humidity: 20 percent to 80 percent

**Power requirements**

► 90 to 126 volts AC;  
50 to 60 hertz

---

**Ordering Information**

**Apple LaserWriter II<sup>mx</sup>**  
Order No. M6215

With your order, you'll receive:  
► LaserWriter II<sup>mx</sup> printer  
► LaserWriter II<sup>mx</sup>/II<sup>mx</sup> Fonts disk

► LaserWriter II Installation disk  
► Letter cassette  
► Toner cartridge  
► Owner's guide  
► Limited warranty statement

**HP LASER JET FONTS**





- LaserJet 500 PLUS: Job offsetting can be used between jobs to allow for easy job separation.

### Printable Characters-Per-Line (Fixed Pitch)

Print Pitch	Portrait/Landscape			
	Std.	Legal	A4	B5
10	80/108	80/138	77/112	68/97
12	96/127	96/163	93/135	80/118
16.66	132/176	132/226	129/188	111/167

### Printable Lines-Per-Page

Lines/Inch	Portrait/Landscape			
	Std.	Legal	A4	B5
8	82/48	80/48	86/48	56/39
8	84/63	108/63	89/81	76/52

### Printable Surface

	Std.	Legal	A4	B5
Width: Inches	8.0	8.0	7.8	6.7
MM	203	203	198	170
Length: Inches	10.8	13.6	11.3	9.7
MM	269	345	287	247

### Power

- Voltage/Frequencies  
115V+/-10% 60Hz
- Optional:  
220V+/-10% 50Hz  
240V+/-10% 50Hz
- Power Consumption at 115 VAC:  
850 Watts Printing Maximum (2900 BTU/Hr)  
170 Watts Standby (580 BTU/Hr)

### Environmental

#### Temperature (Printer & Cartridge)

Operating 10 to 32.5 degrees C.  
(50 to 91 degrees F)

Storage 0 to 35 degrees C.  
(32 to 95 degrees F)

#### Altitude

Operating 0 to 2,500 metres  
(0-8,200 feet)

Non-operating 0 to 15,000 metres  
(0-49,200 feet)

#### Humidity

Operating 20 to 80% RH

Non-Operating 10 to 80% RH

### Audible Noise

Printing < 55 dB(A)

Standby < 45 dB(A)

Note: Measured one meter from source according to ISO/DIS 777

### Physical (LaserJet and LaserJet PLUS)

Width 47.5 cm. (18.5 inches)  
Depth (body only) 41.5 cm. (16.2 inches)  
Depth (with trays) 72.3 cm. (28.2 inches)  
Height 29.3 cm. (11.4 inches)  
Weight 32 kg. (71 pounds)

### Physical (LaserJet 500 PLUS)

Width 47.5 cm. (18.5 inches)  
Depth (body only) 49.5 cm. (19.5 inches)  
Depth (with trays) 87.5 cm. (34.4 inches)  
Height 46.0 cm. (18.1 inches)  
Weight 43 kg. (94 pounds)

## LaserJet Printer Font Product Summary

### Soft Fonts

Ask your HP Dealer or Sales Representative for the latest information on soft fonts. Disc-based soft fonts can be used with the LaserJet PLUS and LaserJet 500 PLUS printers only.

### Font Cartridges (Currently Available)

92286A Courier 1  
92286B Tms Proportional 1  
92286C International 1  
92286D Prestige Elite  
92286E Letter Gothic  
92286F Tms Proportional 2  
92286G Legal Elite  
92286H Legal Courier  
92286J Math Elite  
92286K Math Tms  
92286L Courier Portrait and Landscape  
92286M Prestige Elite Portrait and Landscape  
92286N Letter Gothic Portrait and Landscape  
92286P Tms Roman Portrait and Landscape  
92286Q Memo 1  
92286R Presentations 1  
92286T Tex 1  
92286U Forms Portrait  
92286V Forms Landscape  
92286W Bar Code 3 of 9/OCR-A  
92286X Bar Code EAN/UPC and OCR-B  
92286Y PC Courier 1

For font print samples and other font information, ask your HP Dealer or Sales Representative for the LaserJet Printer Font Cartridge Selection Guide, part number 5954-2270 (or ask to see the LaserJet Printer Family Font Catalog, part no. 5954-2277).

Font cartridges and discs are available through your HP Dealer or through Hewlett-Packard's Direct Marketing Division. Call toll free: 800-538-8787 (in California call 408-738-4133 collect).



**HEWLETT-PACKARD PLOTTER FONTS**





The following table indicates the character sets available on each of the three fonts. The character sets on the same line (such as 0, 10, and 20) have the same characters for each decimal code. These sets are illustrated on the next page.

			Character Set	
17510A/17550A/ 27570A/27580A,B/ 37585A,B/7586B/759XA		27580B/27585B 7586B/759XA	Character Set	ISO Registration Number
Fixed-Space Vector Font	Variable-Space Arc Font	Fixed-Space Arc Font		
0	10	20	ANSI ASCII	006
1	11	21	HP 9825 HPL Character Set	—
2	12	22	French/German	—
3	13	23	Scandinavian	—
4	14	24	Spanish/Latin American	—
5	15	25	Special Symbols	—
6	16	26	JIS ASCII	014
7	17	27	Roman Extensions	—
8	18	28	Katakana	013
9	19	29	ISO IRV (International Reference Version)	002
30	40	50	ISO Swedish	010
31	41	51	ISO Swedish for Names	011
32	42	52	ISO Norwegian Version 1	060
33	43	53	ISO German	021
34	44	54	ISO French	025
35	45	55	ISO British	004
36	46	56	ISO Italian	015
37	47	57	ISO Spanish	017
38	48	58	ISO Portuguese	016
39	49	59	ISO Norwegian Version 2	061
60	70	80	ISO French	069
99	—	—	Drafting	—
100	—	—	Kanji	—
101	—	—	Kanji	—
-1	—	—	Downloadable (user-defined)	—

<sup>1</sup>Character sets 60, 70, 99, 100, 101, and -1 not implemented on the 7510A and 7550A.

<sup>2</sup>Variable-space arc font and character sets 60 and -1 not implemented on the 7570A.

<sup>3</sup>All 7580A/7585A plotters and 7580B/7585B plotters with serial prefix number 2309, 2331, 2334, and 2335 contain only character sets 0-5 and 10-15. 758XB plotters with serial prefix number 2402 and higher contain all character sets (except 39, 60, 100, and 101) in the three fonts.

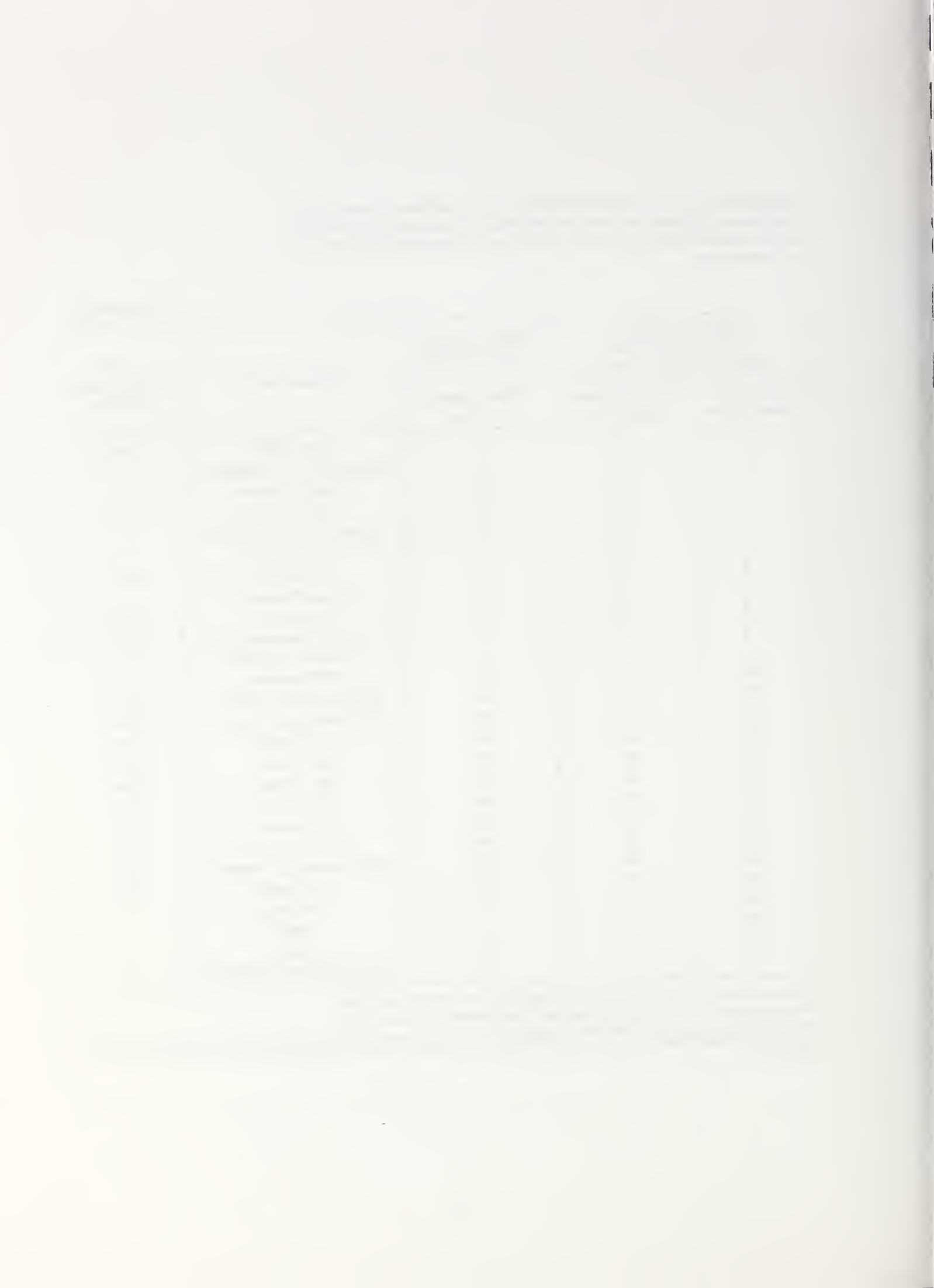


IMAGE 2308 PRINTER FONTS





# Fonts

## TIMES ROMAN

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 abcdefghijklmnopqrstuvwxyz&ÆØøæœßÇçÀàÁáÂâÃãÄäÅåÏïÎîÏï  
 0123456789¼½¾÷-...!?"[]{}^\_`~¡¢£¥¦§¨ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿  
 @©®™\$%&'()\*+,-./:;<=>?@~

## HELVETICA

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 abcdefghijklmnopqrstuvwxyz&ÆØøæœßÇçÀàÁáÂâÃãÄäÅåÏïÎîÏï  
 0123456789¼½¾÷-...!?"[]{}^\_`~¡¢£¥¦§¨ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿  
 @©®™\$%&'()\*+,-./:;<=>?@~

## LUCIDA

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 abcdefghijklmnopqrstuvwxyz&ÆØøæœßÇçÀàÁáÂâÃãÄäÅåÏïÎîÏï  
 0123456789¼½¾÷-...!?"[]{}^\_`~¡¢£¥¦§¨ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿  
 @©®™\$%&'()\*+,-./:;<=>?@~

## LUCIDA SANS

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 abcdefghijklmnopqrstuvwxyz&ÆØøæœßÇçÀàÁáÂâÃãÄäÅåÏïÎîÏï  
 0123456789¼½¾÷-...!?"[]{}^\_`~¡¢£¥¦§¨ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿  
 @©®™\$%&'()\*+,-./:;<=>?@~

## CENTURY SCHOOLBOOK

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 abcdefghijklmnopqrstuvwxyz&ÆØøæœßÇçÀàÁáÂâÃãÄäÅåÏïÎîÏï  
 0123456789¼½¾÷-...!?"[]{}^\_`~¡¢£¥¦§¨ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿  
 @©®™\$%&'()\*+,-./:;<=>?@~

## Proprietary Custom Fonts—Optional

Point Sizes	6	7	8	9	10	11	12	14	16	18	20	22	24	28	36
LUCIDA™ Roman, Bold, Italic 158 Characters in set	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
LUCIDA SANS™ Roman, Bold, Italic 161 Characters in set	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
HELVETICA® Roman, Bold, Italic 161 Characters in set	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
TIMES ROMAN® Roman, Bold, Italic 161 Characters in set	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
CENTURY SCHOOLBOOK® Roman, Bold, Italic, Bold Italic 161 Characters in set	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
PI SYMBOLS 116 Characters in set	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

\*36pt available in ROMAN only

## Standard Fonts

Courier 20 pitch  
 Courier 15 pitch  
 Courier 12 pitch  
 Courier 10 pitch  
 Courier 8 pitch  
 Courier Bold 12 pitch  
 Courier Bold 10 pitch

Courier Character set: 177

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 abcdefghijklmnopqrstuvwxyz  
 0123456789\$%&'()\*+,-./:;<=>?@~

...!?"[]{}^\_`~¡¢£¥¦§¨ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿  
 @©®™\$%&'()\*+,-./:;<=>?@~  
 AAÆEËÎÑÓÔÕÖÙÀÁÂÃÄÅæßàáâãäåäöüü

## Downloadable Fonts Available at No Cost

Our UNIX support tape provides 17 complete sets of public domain fonts which have 15 different point sizes (6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 28, 36). We also provide a number of partial sets for over 890 different font files.

Our VAX/VMS T<sub>E</sub>X support tape provides over 70 different Metafont fonts which allow you to generate additional sizes and fonts.



QMS-PS 800 FONTS





# THE QMS-PS 800

**PostScript**—PostScript is a page description language that is compact, universal, convenient, portable and flexible. It differs in a very fundamental way from all other page description languages. Namely, a PostScript description of a page is an executable program in a graphics programming language, rather than being data input to some page formatter. Like any complete programming language, PostScript has operators and operands, variables and constants, a syntax and a semantics, and a customary usage style. Unlike most programming languages, it also has powerful graphics primitives that underlie a uniform and consistent, device-independent, graphics-imaging model.

**Transformations on Text and Graphics**—QMS-PS 800 allows text and graphics to be scaled and rotated. The full generality of two-dimensional matrix transformations is available. Scale, rotate and translate primitives exist in the language, but other transformation specifications can be built up.

**Synthetic Halftone Text and Graphics**—QMS-PS 800 provides capabilities for synthetic halftone text and graphics. Graphic shapes and text can be filled with halftone color or gray values. All parameters of the halftone process (spot shape, orientation, screen frequency and transfer function) may be specified by the user.

**Arbitrary Clipping of Text and Graphics**—With the QMS-PS 800, the user can specify arbitrary clipping boundaries for text and graphics. PostScript can crop any text, graphics or scanned image to its clipping region.

---

## TYPEFACES

With the QMS-PS 800, you have typefaces that include: Times, Times Bold, Times Bold Italic, Helvetica, Helvetica Bold, Helvetica Oblique, Helvetica Bold Oblique, Courier, Courier Bold, Courier Oblique, Courier Bold Oblique, as well as Symbol, Underline, Shadow and Hollow Styles for all resident typefaces. They're scaleable from four points upward and can be rotated to any orientation.

---

## GRAPHICS

Generate circles, graphs, pie and bar charts, letters, logos, and transparencies for business. Intricate three-dimensional perspective plots for engineering. High-quality forms and documents with multiple typefaces. And phototypesetting simulations for electronic publishing.

## PRINT SPEED

Eight pages per minute. Actual printing speed depends on application.

## PRINTER SPECIFICATIONS

**TYPE**—Desk-top, electronic page printer.

**PRINT METHOD**—Raster scan semiconductor diode laser beam with rotating polygon mirror and lens deflection onto photoreceptor; xerographic image transfer to plain paper.

**PRINT SPEED**—Eight pages per minute. Actual printing speed depends on application.

**LASER**—Semiconductor laser scanning system.

**TONER**—Dry, monocomponent in user-replaceable cartridges.

**PAPER**—8½" x 11"; 8½" x 14"; A4 (international); B5 (international); envelopes, charts, transparencies (manual feed); minimum weight—16 lbs.; Maximum weight—24 lbs.

**PAPER LOADING**—Cassette, 100 sheets each.

**DOT PITCH**—300 dots per inch horizontal, 300 dots per inch vertical.

**WARM-UP TIME**—Two minutes from cold start.

**ELECTRICAL REQUIREMENTS**—Printing-0.8 KVA; Standby-0.3 KVA

**OPERATING ENVIRONMENT**—50-90°F (10-30°C); Humidity 20-80%

**RECOMMENDED USE LEVEL**—Up to 3,000 pages per month.

**DIMENSIONS**—18.7" (W) x 11.4" (H) x 16.3" (D).

**WEIGHT**—65 lbs.

**NOISE LEVEL**—Less than 55 dBA.

**STANDARD HOST INTERFACES**—RS232, RS422 and AppleTalk™ port.

**MEMORY REQUIREMENTS**—2M RAM and .5M ROM.



## BITSTREAM FONTWARE







# Typeface Package Order Form

(Please Print)

Name

Title Telephone

Company

Address

City, State, Zip Code Country

Ship To (if different from above)

Name

Address

City, State, Zip Code Country

## To Order

Please complete this form and  
mail to Bitstream Inc.

Or...

Order by Phone: 800-522-3668

In Massachusetts: 617-497-7512

Order No.	Typeface Package Name	Quantity	Price: \$195.00 each
FTP-A1	Swiss		
FTP-A2	Century Schoolbook		
FTP-A3	Dutch		
FTP-A4	Zapf Calligraphic		
FTP-A5	Futura Light		
FTP-A6	Swiss Light		
FTP-A7	Swiss Condensed		
FTP-A8	Futura Book		
FTP-A9	Futura Medium		
FTP-A10	Courier (10 Pitch)		
FTP-A11	Letter Gothic (12 Pitch)		
FTP-A12	Prestige (12 Pitch)		
FTP-A13	ITC Avant Garde		
FTP-A14	Zapf Humanist		
FTP-A15	ITC Garamond		
FTP-A16	ITC Souvenir		
FTP-A17	ITC Korinna		
FTP-A18	Bitstream Charter		
FTP-A19	ITC Galliard		
FTP-A20	Headlines 1: Bitstream Cooper Black • Broadway University Roman • Cloister Black		

SEYBOLD,  
SANTA CLARA,  
CALIFORNIA.  
SEPTEMBER  
9-12 1987.

# Bitstream Fontware

## Fontware Typefaces

Swiss  
Century Schoolbook  
Dutch  
Zapf Calligraphic  
Futura Light  
Swiss Light  
Swiss Condensed  
Futura Book  
Futura Medium  
Courier (10 pitch)  
Letter Gothic (12 pitch)  
Prestige (12 pitch)  
ITC Avant Garde  
Zapf Humanist  
ITC Garamond  
ITC Souvenir  
ITC Korinna  
Bitstream Charter  
ITC Galliard  
Headline 1:  
Cooper Black  
Broadway  
University Roman  
Cloister Black

## About Bitstream

Based in Cambridge, Massachusetts, Bitstream was the first independent digital type foundry. Founded by a prestigious group of typographic professionals, Bitstream has grown rapidly to a team of 75 expert type designers, software engineers, as well as marketing and customer support professionals.

An industry leader in typographic quality and innovative technology, Bitstream supplies digital type and related software to 200 equipment manufacturers and software developers throughout the world. That means you see Bitstream type every day, everywhere: in newspapers, magazines, books, television, video and business documents.

## For More Information:

Bitstream sales will assist you in getting the typeface of your choice, compatible with your desktop publishing system. Call Bitstream at 800-522-3668.

## Fontware for Your PC

Fontware is the first program that makes fonts on the IBM PC/AT or compatibles in virtually any size for most popular displays and printers from typeface packages comprising the Fontware Library. Fontware Installation Kits are available for Ventura Publisher and for Microsoft Windows applications, among more to come.

Fontware gives you professional typographic quality and variety so you can make the most of desktop publishing. And since Fontware creates matching screen and printer fonts, you can preview your document accurately before you print it.

## Industry Leaders Endorse Fontware

Paul Brainerd, President of Aldus Corporation said, "Having been a part of the evolution of Fontware over the last 18 months we know that the font requirements of our PC PageMaker customers will be realized thanks to Fontware." Bitstream's new product allows for matching screen and printer fonts and complete control over type fonts and character adjustments, which was not available for the PC until Fontware.

Ventura Software has been working with Bitstream to create Fontware for Ventura Publisher. "The result is a product which provides exactly the capabilities that desktop publishing users have been asking for: freedom to choose any type size, true WYSIWYG capability, device independence and a large, professional quality type library," said John H. Meyer, President of Ventura Software, Inc.

*This newsletter was produced using electronic publishing techniques. It was set in Swiss Black Condensed, Swiss Condensed, Bitstream Charter, and Swiss Bold Italic. All fonts were generated by the Fontware Installation Kit for Microsoft Windows using Fontware Outlines.*

**HEWLETT-PACKARD PRINTER FONTS**





### 3 Selecting and Controlling Fonts

Hewlett-Packard has standardized the typefaces in Figure 3.4:

<u>Value of T</u>	<u>Typeface</u>
0	Line Printer
1	Pica
2	Elite
3	Courier
4	Helvetica (HELV, Swiss)
5	Times Roman, Dutch (TMS RMN)
6	Gothic
7	Script
8	Prestige
9	Caslon
10	Orator
11	Presentation
12	Helvetica Condensed
14	Futura
15	Palatino (Zapf Calligraphic)
16	Souvenir
17	Zapf Humanist (Optima)
18	Garamond
19	Cooper
20	Coronet
21	Broadway
22	Bodoni
23	Century Schoolbook
24	University Roman
25	Avant Garde Gothic
27	Korinna
28	Bitstream Charter
29	Cloister Black
30	Galliard
136	Futura Book
146	Futura Light
148	Helvetica Light

Figure 3.4      *Hewlett-Packard Typeface Designations*



## GENIGRAPHIC FONTS





## Appendix B – Font List

---

Font Abbreviation	Index	Bitstream Name
FONT1	001	(like) Gothic 731 Bold Gothic
FONT2	002	(like) Swiss 721 Bold (Helvetica Medium)
FONT3	003	(like) Clarendon 701 Clarendon Light [Craw Clarendon]
FONT4	004	(like) Swiss 721 Roman [Helvetica]
SWSR	006	Swiss 721 Roman
SWSI	007	Swiss 721 Italic
GOTBC	008	Gothic 731 Bold Condensed
SWSB	010	Swiss 721 Bold
DUTR	011	Dutch 801 Roman
DUTI	012	Dutch 801 Italic
SQRBX	013	Square 721 Bold Extended
SQRX	014	Square 721 Extended
CLRL	015	Clarendon 701 Clarendon Light
BODB	017	Modern 421 Bodoni Bold
BODBI	018	Modern 421 Bodoni Bold Italic
SWSC	019	Swiss 742 Condensed
SWSBC	020	Swiss 722 Bold Condensed
SWBU	021	Swiss 742 Bold
SWBL	022	Swiss 742 Light
GMFL	023	Geometric 211 Futura Light
GMFH	024	Geometric 211 Futura Heavy
ZHDI	025	Zapf Humanist 601 Demi
SWSH	026	Swiss 721 Heavy
SWSBK	027	Swiss 721 Black
FRTZQ	028	Flareserif 861 Friz Quadrata
FRZQB	029	Flareserif 861 Friz Quadrata Bold
CNSCH	030	Century 702 Century Schoolbook
CSCHB	031	Century 702 Century Schoolbook Bold
ZHDIB	032	Zapf Humanist 601 Bold
SWSBO	033	Swiss 721 Bold Outline
BRSH	034	Brush 451 Brush Script
GARL	037	Aldine 851 ITC Garamond Light
GARB	038	Aldine 851 ITC Garamond Book Condensed
GARBD	039	Aldine 851 ITC Garamond Bold Condensed
PI2	098	Pi Font 2



ADOBE FONT METRIC FILES







**ADOBE® FONT METRIC FILES  
Specification  
Version 2.0**

**January 16, 1989  
PostScript® Developer Tools & Strategies Group**

**Adobe Systems Incorporated  
1585 Charleston Road PO Box 7900  
Mountain View, CA 94039-7900  
(415) 961-4400**

Copyright © 1989, 1988, 1987 by Adobe Systems Incorporated.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

PostScript and Adobe are registered trademarks of and the PostScript logo is a trademark of Adobe Systems Incorporated.

The information herein is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in this book. The software described in this book is furnished under license and may only be used or copied in accordance with the terms of such license.



# ADOBE FONT METRIC FILES Specification Version 2.0

January 16, 1989

PostScript® Developer Tools and Strategies Group

## 1. INTRODUCTION

This document describes a standard interchange format for communicating font metric information to people and programs. The format is ASCII encoded (for both human and machine readability), machine independent, and extensible. Files in this format are known as Adobe® Font Metrics (AFM) files and are available for all of Adobe Systems's PostScript fonts.

## 2. PARSING DETAILS

Each AFM file contains the information for one PostScript printer font. The file begins with global information pertaining to the font as a whole, followed by sections with character metrics, kerning data (optional) and composite character data (optional). The file format is line-oriented, each line beginning with a property (key) name, followed by the values for that property. Keys and values are separated by one or more white space characters (space or tab).

The format is:

**Key value value ...**

Key names are case-sensitive. All keys beginning with a capital letter are reserved for use by Adobe Systems; user-defined non-standard entries should begin with a lowercase letter. The Adobe Systems standard keys are detailed below, but other keys are allowed and should be ignored by programs not recognizing them.

Values will be one of the following: **string, name, number, integer or boolean.**

- Strings are terminated by the end of line.
- Names are similar to strings except that they may not contain any white space characters; they are terminated by white space characters or by a special termination character (see section on "Character Metrics" below).
- Numbers, integers and booleans are separated by white space characters, which may be space, newline, or tab.
- A number can be either a real number or an integer, and both must be allowed for.
- A boolean is either the value **true** or **false**.



## 2.1 COMMENTS

Comments may be present in a font metric file. They are introduced by the keyword **Comment**, and are terminated by the end of line. Lines should be no longer than 255 characters under any circumstances.

### **Comment string**

The text is arbitrary, and should be ignored.

## 3. FILE STRUCTURE

An AFM file has several sections, each of which is delimited by a **Start** and **End** keyword. The main section contains a "header" of global font information, and each of the other sections of the file is hierarchically one level down from the main section.

### **StartFontMetrics version** **EndFontMetrics**

These comments delimit the entire font metrics file. The **StartFontMetrics** line should be the first line in the file, and the **EndFontMetrics** keyword should be the last line in the file.

## 3.1 GLOBAL FONT INFORMATION

The following global font keys are the same as those in the top level or **FontInfo** subdictionary of an Adobe Systems PostScript font dictionary. Their meanings are described in the chapter on fonts in the *PostScript Language Reference Manual*. Please note that although some of the keys in the PostScript **FontInfo** subdictionary begin with a lowercase letter (e.g., **isFixedPitch**, **version**) all keys listed here begin with uppercase letters to distinguish them as keys reserved for use by Adobe Systems. All numeric values are in the character coordinate system (1000 units per em).

### **FontName string**

Name of the font as presented to the PostScript **findfont** operator.

*Example:* Garamond-Light.

### **FullName string**

The full text name of the font. *Example:* ITC Garamond Light.

### **FamilyName string**

The name of the "font family" to which the font belongs.

*Example:* ITC Garamond.

### **Weight string**

Weight of the font. *Example:* Roman, Bold, Light, etc.

**ItalicAngle number**

Angle (in degrees counter-clockwise from the vertical) of the dominant vertical strokes of the font.

*Example: -12.*

**IsFixedPitch boolean**

If *boolean* is true, this indicates that the font is a fixed pitch (monospaced) font. A value of false indicates a proportionally spaced font.

**FontBBox lly lly urx ury**

Four *numbers* giving the lower left corner and the upper right corner of the font bounding box. *Note:* the bounding box given here is that of the flattened paths, not the Bezier curve descriptions. These values are all *integers*, and should be rounded off if necessary.

**UnderlinePosition number**

Distance from the baseline for centering underlining strokes.

**UnderlineThickness number**

This is the stroke width for underlining, expressed in character coordinate space (proportional to the font characters). It should be adjusted for point size by any application wishing to perform underlining.

**Version string**

Font version identifier. Matches the string found in the FontInfo dictionary of the font itself.

**Notice string**

Font name trademark or copyright notice.

**EncodingScheme string**

String indicating the default encoding vector for this font. The most common one is *AdobeStandardEncoding*. Special fonts may simply state *FontSpecific*.

**CapHeight number**

Top of capital H, measured in character coordinate system.

**XHeight number**

Top of lower case x, measured in character coordinates.

**Ascender number**

Top of lower case d.

**Descender number**

Bottom of lower case p.

## 4. INDIVIDUAL CHARACTER METRICS

Each character's metrics consists of a list of keys and values separated by semicolons; the metrics for a given character will always be contained in one line. The characters are sorted by numerically ascending character code. Unencoded characters follow the encoded characters and are identified by character codes of -1.

*Example:* A character metric data line might look like this:

**C 102; WX 333; N f; B 21 0 382 685; L l fi; L l fi;**

**StartCharMetrics** *integer*

**EndCharMetrics**

These comments introduce (and conclude) the character metrics section of the file. The *integer* value indicates how many individual characters to expect.

**C** *integer*

Decimal value of default PostScript character code (-1 if unencoded).

**WX** *number*

Character width in x (y is 0).

**W** *number<sub>x</sub> number<sub>y</sub>*

Character width vector (x,y)

**N** *name*

PostScript character name.

**B** *llx lly urx ury*

Character bounding box where *llx*, *lly*, *urx*, and *ury* are all *numbers*.

**L** *successor ligature*

Ligature sequence where *successor* and *ligature* are both *names*. The current character may join with the character named *successor* to form the character named *ligature*. Note that characters may have more than one such entry. (See example above.)

## 5. KERNING DATA

The kerning data section is optional; it may or may not be present for a given font. The section is surrounded by the lines **StartKernData** and **EndKernData**. Kerning data is supplied in two forms: track kerning and pair-wise kerning. Track kerning is applied to all characters uniformly whereas pair-wise kerning is applied to specific character pairs. Track kerning and pair-wise kerning may be used independently or together (i.e., it is possible to apply track kerning to a line of text and then to apply pair-wise kerning on top of that). The two forms of kerning data are treated as subsections within the kerning data section and both sections need not be present.



StartKernData  
EndKernData

These comments introduce (and conclude) the kerning section of the file.

## 5.1 TRACK KERNING

The track kerning data is surrounded by the lines:

StartTrackKern *integer*  
EndTrackKern

Where *integer* indicates how many different sets of track kerning data are present.

Normally track kerning is provided in different *degrees* of tightness. Within a track (a degree of tightness), the amount to decrease (or possibly increase) the amount of space between characters increases (or possibly decreases) with the point size of the font (e.g., for tight track kerning, the amount to decrease the space between characters at 6 point might be 0.1 points and at 72 point it might be 3.78 points).

The data itself begins with the key TrackKern and is followed by the track kerning information:

TrackKern *degree min-ptsize min-kern max-ptsize max-kern*

The *degree* is an integer where increasingly negative degrees represent tighter track kerning and increasingly positive degrees represent looser track kerning. *min-pt-size*, *min-kern-amt*, *max-pt-size* and *max-kern-amt* are all *numbers*. Since the track kerning is a linear function, the minimum and maximum cut-off values (point sizes) are provided along with the amount to track kern by at the point size. The kerning amounts are given relative to the point size. From those 4 values, the track kerning function can be derived. The track kerning function is a linear function. The equation for the line can be determined from the data provided and, therefore, the track kerning values for any point size can be determined. The track kerning values for any point size below/above the minimum/maximum point size are constant (the minimum kerning amount/maximum kerning amount).

In general the track kerning function is as follows:

TrackKern *degree p<sub>0</sub> k<sub>0</sub> p<sub>1</sub> k<sub>1</sub>*

Where *x* = current point size

Where *k<sub>1</sub>* = max-kern-amt, *k<sub>0</sub>* = min-kern-amt

Where *p<sub>1</sub>* = max-pt-size, *p<sub>0</sub>* = min-pt-size

$$\begin{aligned} f(x) &= k_0 && \text{for } x < p_0 \\ f(x) &= \frac{(k_1 - k_0) \cdot x + (k_0 \cdot p_1 - k_1 \cdot p_0)}{(p_1 - p_0)} && \text{for } p_0 \leq x \leq p_1 \\ f(x) &= k_1 && \text{for } x > p_1 \end{aligned}$$



See the last section of this document for a good example of these keywords in use.  
Below is a sample of text printed using these track kerning values.

**Figure 1: Track Kerning**

<hr/>	
6 pt	
no kerning	An illustration of how track kerning works.
light kerning	An illustration of how track kerning works.
medium kerning	An illustration of how track kerning works.
tight kerning	An illustration of how track kerning works.
<hr/>	
12 pt	
no kerning	An illustration of how track kerning works.
light kerning	An illustration of how track kerning works.
medium kerning	An illustration of how track kerning works.
tight kerning	An illustration of how track kerning works.
<hr/>	
18 pt	
no kerning	An illustration of how track kerning works.
light kerning	An illustration of how track kerning works.
medium kerning	An illustration of how track kerning works.
tight kerning	An illustration of how track kerning works.

## 5.2 PAIR-WISE KERNING

The pair-wise kerning data is surrounded by the lines:

**StartKernPairs** *integer*  
**EndKernPairs**

Where *integer* indicates how many pairs to expect.

There will be one kerning pair per line. Each line will begin with a keyword of the form KP or KPX.

**KP** *name<sub>1</sub> name<sub>2</sub> number<sub>x</sub> number<sub>y</sub>*

Name of the first character in the kerning pair followed by the name of the second character followed by the kerning vector specified as an (x,y) pair. The kerning vector is the amount to move the second character by relative to the first character to position it properly. The kerning vector is specified in the character coordinate system. In order to use this vector it is necessary to transform it into user space and scale it by the point size in use. The best way to do this is to use the FontMatrix entry in the current font dictionary.

**KPX** *name<sub>1</sub> name<sub>2</sub> number<sub>x</sub>*

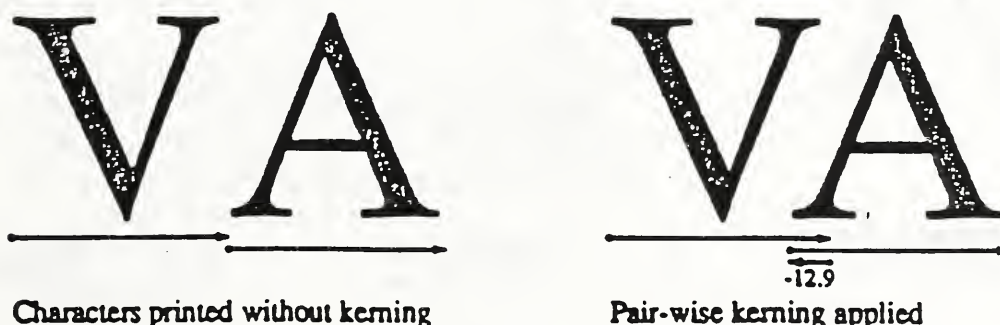
Name of the first character in the kerning pair followed by the name of the second character followed by the kerning amount in the x direction (y is zero). The kerning amount is specified in the units of the character coordinate system.

A character pair kerning line might look like this:

KPX V A -129

Below is an example of pair-wise kerning applied to 100 point characters:

**Figure 2: Pair-wise kerning**



## 6. COMPOSITE CHARACTER DATA

The composite character data section is also optional. Composite characters are new characters that are made up of characters already existing in the font, such as accented characters. Character metric information for composite characters is found in the Character Metrics section of the AFM file. Although most PostScript fonts available from Adobe Systems include a rather extensive set of composite characters, some applications may wish to generate their own. This section provides the data necessary for accurate positioning of the individual pieces. All units are expressed in the 1000 unit-per-em character coordinate system.

**StartComposites** *integer*  
**EndComposites**

Where *integer* indicates how many pairs to expect.

The data for each composite character is represented as a list of keys and values separated by semicolons. Each composite character gets one line of description. The standard keys are:

**CC** *name integer*

The composite character name followed by the number of parts that make up the composite.

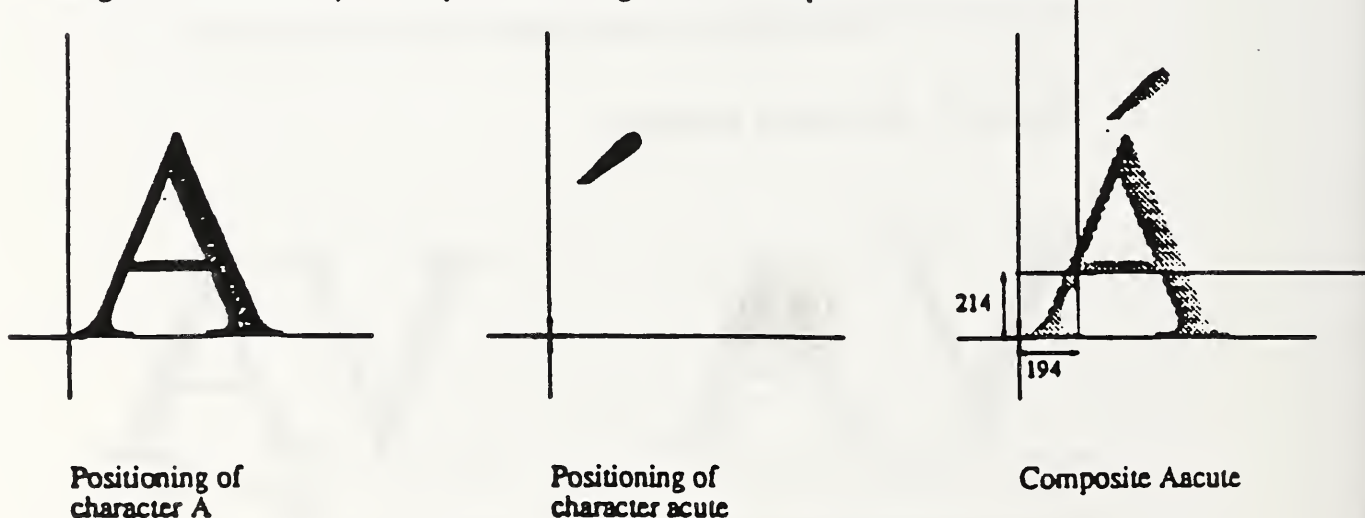
**PCC** *name deltax deltay*

One of the parts of the composite character. The character name is given followed by the x and y displacement from the origin.

A composite character line might look like this:

CC Aacute 2; PCC A 0 0; PCC acute 194 214;

Figure 3: Example of positioning for a composite character



## 7. EXAMPLE FILE

The following is an example of an AFM file for Times Roman, although some of the data have been omitted to keep it short.

```
StartFontMetrics 2.0
Comment Copyright (c) 1984 Adobe Systems Incorporated.
Comment All Rights Reserved.
FontName Times-Roman
FullName Times Roman
FamilyName Times
Weight Medium
ItalicAngle 0.0
IsFixedPitch false
UnderlinePosition -98
UnderlineThickness 54
Version 001.000
Notice Times is a trademark of Allied Corporation.
EncodingScheme AdobeStandardEncoding
FontBBox -167 -252 1004 904
CapHeight 673
XHeight 445
Descender -219
Ascender 686
StartCharMetrics 210
C 32 ; WX 250 ; N space ; B 0 0 0 0 ;
C 33 ; WX 333 ; N exclam ; B 128 -17 240 673 ;
C 34 ; WX 408 ; N quotedbl ; B 46 445 313 685 ;
C 35 ; WX 500 ; N numbersign ; B 20 -17 481 673 ;
C 36 ; WX 500 ; N dollar ; B 45 -92 456 726 ;
C 37 ; WX 833 ; N percent ; B 63 -36 771 655 ;
... - lines omitted for brevity -
C 101 ; WX 444 ; N e ; B 24 -17 416 469 ;
C 102 ; WX 333 ; N i ; B 21 0 382 685 ; L i fi ; L i fl ;
C 103 ; WX 500 ; N g ; B 24 -220 469 468 ;
C 104 ; WX 500 ; N h ; B 11 0 487 686 ;
C 105 ; WX 278 ; N l ; B 25 0 255 685 ;
C 106 ; WX 278 ; N j ; B -56 -217 205 685 ;
C 107 ; WX 500 ; N k ; B 7 0 496 686 ;
... - lines omitted for brevity -
C 248 ; WX 278 ; N slash ; B 0 0 283 685 ;
C 249 ; WX 500 ; N slash ; B 30 -104 469 566 ;
C 250 ; WX 722 ; N oe ; B 30 -10 684 462 ;
C 251 ; WX 500 ; N germandbls ; B 13 0 468 686 ;
C -1 ; WX 722 ; N Aacute ; B 22 0 702 873 ;
C -1 ; WX 722 ; N Acircumflex ; B 22 0 702 875 ;
C -1 ; WX 722 ; N Adieresis ; B 22 0 702 819 ;
... - lines omitted for brevity -
EndCharMetrics
StartKernData
StartTrackKern 3
```



Comment Light kerning  
TrackKern -1 14 0 72 -1.89  
Comment Medium kerning  
TrackKern -2 8 0 72 -3.2  
Comment Tight kerning  
TrackKern -3 6 -.1 72 -3.78  
EndTrackKern  
StartKernPairs 2  
KPX V A -129  
KPX A Y -92  
EndKernPairs  
EndKernData  
StartComposites 1  
CC Aacute 2; PCC A 0 0; PCC acute 194 214;  
EndComposites  
EndFontMetrics

**APPENDIX C**  
**REGISTRATION PROPOSALS**



PART 1

PROPOSALS APPROVED BY X3H3 IN SEPTEMBER 1989  
IN LETTER BALLOT #80. FINAL TEXT READY TO  
FORWARD TO ISO IS INCLUDED.





**Proposal Number:** 1

**Presentation date of proposal:** 10 April 1987

**Sponsoring Authority:** ANSI

**Class of Graphical Item:** LINETYPE

**Name:** user-specified dash pattern

#### Description

This user-specified dash pattern linetype consists of alternating dashes and spaces. By using the registered escape function Set Dash, a user can exercise precise control over important aspects of the appearance of lines of this type. This control includes the ability to select the length of each dash and of each space, the offset to the start of the "dash pattern", and whether the dash pattern is restarted at each portion of a primitive. In addition, line cap, line join, and mitre limit—each of which can be set by a registered escape function—apply to primitives of this linetype.

#### Additional Comments

This registration proposal is accompanied by a proposal to register an escape function - Set Dash - that defines the current user-specified dash pattern. It is intended that these proposals be processed together.

This linetype is not intended to be used as an edgetype.

#### Justification for Inclusion

User specified linetypes are needed to support the requirements of office document exchange and publishing. They are commonly found in widely available proprietary graphics systems.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered linetype to supplement those defined in 5.4.1.
- 2) ISO 8632 (CGM) - Specifies a registered linetype to supplement those defined in 5.7.2.
- 3) ANSI Y14.2M-1979 - Line Conventions and Lettering.

<b>Date of Presentation:</b> 10 April 1987
--

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> ESCAPE
--

<b>Specific Escape Function Identifier:</b> Set Dash
--

Description
-------------

This escape function sets the characteristics of the user-specified (registered) linetype. Such a user-specified dash pattern linetype consists of alternating dashes and spaces. This escape function allows a user to exercise precise control over important aspects of the appearance of lines of this type. This control includes the ability to select the length of each dash and of each space, the offset to the start of the "dash pattern", and whether the dash pattern is restarted at each portion of a primitive. In addition, line cap, line join, and mitre limit—each of which can be set by a registered escape function—apply to primitives of this linetype. See attached sheet for additional details.

Additional Comments
---------------------

None.

Justification for Inclusion
-----------------------------

User specified line types are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing.

Relationship to Standards
---------------------------

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

Description:

**Set Dash** controls the dash pattern used with line primitives of registered linetype "user-specified dash pattern" (linetype TBD). If the array of dash pattern lengths is empty (i.e., the number of lengths is zero), the linetype is equivalent to solid. This is the default value. If the array of dash pattern lengths is not empty, line primitives of registered linetype "user-specified dash pattern" are drawn with dashes whose pattern is defined by the array of lengths.

Each length in the array of dash pattern lengths must be non-negative. If any length in the dash pattern length array is negative, a length of zero shall be substituted. At least one length in the array must be non-zero. If all lengths are zero, the linetype is equivalent to solid. If the number of lengths is less than zero, a value of zero shall be substituted.

The elements of the array of dash pattern lengths are interpreted in sequence as relative distances along the primitive. These distances alternately specify the length of a gap between dashes. The contents of the array are used cyclically, that is when the end of the array is reached, the pattern starts over at the beginning.

Dashed lines wrap around curves and corners just as solid lines do. The ends of each dash receive no special treatments. In particular, the "ends" of dashes are not treated with current line cap. No measures other than continuity as described below, are provided to coordinate the dash pattern with features of an output primitive.

The *offset* value may be thought of as the "phase" of the dash pattern relative to the start of the path. It is interpreted as a distance into the dash pattern at which the pattern should be started. Before beginning output of the dash pattern, the elements of the array of dash pattern lengths are cycled through, and the distances of alternating dashes and gaps added up, but without generating any output. When the offset distance into dash pattern has been reached, the primitive is drawn (from its beginning) using the dash pattern from the point that has been reached. If the offset is greater than the total length (the sum of all lengths in the dash pattern length array), the lengths in the array shall be re-cycled from the beginning. This process shall be repeated as many times as necessary until the offset distance is reached.

When *continuity* is set to restart, each portion of a primitive (e.g. each line segment within a polyline) is treated independently; i.e. the dash pattern is restarted (and offset applied) at the beginning of each portion. When *continuity* is set to continuous, the dash pattern is not restarted in going from one portion of a primitive to the next. If *continuity* is not either restart or continuous, the default value of restart shall be used.



Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

The elements of the array of dash pattern lengths are interpreted in sequence as distances in VDC units along the primitive. The offset value is in VDC units. A functional description of the Set Dash escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):

offset (VDC)

continuity (one of: restart, continuous) (E)

list of lengths (nVDC)

dash pattern lengths array

Items for Data Record:

offset

continuity

number of lengths

dash pattern lengths array

Data Record Description:

The parameters define the offset, continuity, number of dash pattern lengths, and the dash pattern lengths.

## Set Dash

### 2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

The set dash escape is applicable at GKS level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type	Range
escape function identifier	as assigned	N	
input data record:			
offset	WC	R	
continuity (RESTART, CONTINUOUS)		E	
number of lengths		I	>=1
dash pattern lengths			
array	WC	nxR	
output data record:			
none			

#### Errors:

- 8 GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP

## Set Dash

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqr" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqr is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqr(CONT, DIMLEN, OFFSET, LEN)
```

#### Input Parameters:

INTEGER CONT	continuity (RESTART, CONTINUOUS)
INTEGER DIMLEN	dimension of the dash pattern lengths array
REAL OFFSET	offset into dash pattern
REAL LEN(DIMLEN)	dash pattern lengths array

#### Output Parameters:

NONE

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

continuity indicator	restart,	continuous
INTEGER	GREST,	GCONT
PARAMETER	(GREST=0,	GCONT=1)

b) The following parameters are proposed for use when accessing this escape through the GESC function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

INTEGER IL	2
INTEGER IA( 1 )	continuity (RESTART, CONTINUOUS)
INTEGER IA( 2 )	number of dash pattern lengths
INTEGER RL	1+ number of dash pattern lengths
REAL RA( 1 )	offset
REAL RA( 2 )	first length
REAL RA( 3 )	second length
...	
REAL RA(1 + number of dash pattern lengths)	last length
INTEGER SL	0

The Unpack Data Record function is not required by this escape.

5) Pascal language binding (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

GEscapeContinuity = (GVEscapeRestart, GVEscapeContinuous);

```

GEscapeDataIn = RECORD
    CASE EscapeId : GTEscapeDataTag of
        1: (
            U0001 Offset          : REAL
            U0001 Continuity      : GEscapeContinuity;
            U0001 NumberLengths   : INTEGER;
            U0001 Lengths         : REAL );
    END;

```

```

GEscapeDataOut = RECORD
    CASE EscapeID : GTEscapeDataTag of
        1: ( ) ;
    END;
    (*Null Record*)

```



## Set Dash

6) GKS Ada language binding (reference ISO/IEC 8651-3 GKS Language Bindings; Part 3: Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SET\_DASH" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a user specified dash pattern.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type OFFSET is ESCAPE_FLOAT;
  type CONTINUITY_CHOICE is (RESTART, CONTINUOUS);
  type DASH_PATTERN_LENGTHS_ARRAY is array
    (SMALL_NATURAL range <>) of ESCAPE_FLOAT;
  type SET_DASH_DATA_RECORD (NUMBER_OF_LENGTHS: SMALL_NATURAL := ()) is
    record
      OFFSET                                :in ESCAPE_FLOAT;
      CONTINUITY                           :in CONTINUITY_CHOICE;
      DASH_PATTERN_LENGTHS                 :in DASH_PATTERN_LENGTH_ARRAY
                                         (1..NUMBER_OF_LENGTHS);
    end record;

  procedure SET_DASH (DASH_RECORD :in SET_DASH_DATA_RECORD);

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

Proposal Number: 37

Date of Presentation: 10 April 1987

Sponsoring Authority: ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Set Line Cap

#### Description

This escape function sets a value for the current line cap. This value determines the shape put at the ends of line segments. The shape is not applied to the ends of individual "dashes" that compose some linetypes. See attached sheet for additional details.

#### Additional Comments

None.

#### Justification for Inclusion

User specified line caps are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

Description:

**Set Line Cap** selects the line cap value specified by the *line cap indicator*. This value determines the shape to be put at the ends of line segments. The shape is not applied to the ends of individual "dashes" that compose some linetypes. The following line cap values are defined with this proposal:

**butt cap** : the line is squared off at the endpoint; there is no projection beyond the endpoint.

**round cap** : a semicircular arc with diameter equal to the line width is drawn around the endpoint and filled. The drawn line thus projects beyond the endpoint.

**projecting square cap** : the line is squared off at a distance equal to half the line width beyond the endpoint.

If the line cap value is not one of the defined values, the default value, butt cap, shall be used.

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Line Cap escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):  
line cap indicator (E)

Items for Data Record:

line cap indicator

The following line cap indicator values are defined:

- 1: butt cap
- 2: round cap
- 3: projecting square cap

Data Record Description:

The parameter defines the line cap value.

## 2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

## 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
line cap indicator	(BUTT, ROUND, PROJECTING SQUARE)	E
output data record:		
none		

### Errors:

- 8 GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP

## 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrs(LNCAP)

Input Parameters:

INTEGER LNCAP                      line cap indicator (BUTT, ROUND, PROJECTING SQUARE)

Output Parameters:

NONE



### Set Line Cap

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

line cap indicator	butt,	round,	projecting square
INTEGER	GBUTT,	GROUND,	GPROSQ
PARAMETER	(GBUTT=1,	GROUND=2,	GPROSQ=3)

b) The following parameters are proposed for use when accessing this escape through the GESC function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

INTEGER IL	1
INTEGER IA (1)	line cap indicator (BUTT, ROUND, PROJECTING SQUARE)
INTEGER RL 0	
INTEGER SL 0	

The Unpack Data Record function is not required by this escape.

5) **Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```

GEscapeLineEndType = (GVEscapeButt, GVEscapeRound,
                      GVEscapeProjectingSquare);

GREScapeDataIn = RECORD
    CASE EscapeID : GTEscapeDataTag of
        1: (
            U0001 Linecap : GEscapeLineEndType);
    End;

GREScapeDataOut = RECORD
    CASE EscapeId  : GTEscapeDataTag of
        1: ( ) ; (*Null Record *)
    END;

```

## Set Line Cap

6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SET\_LINE\_CAP" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set line cap.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type LINE_CAP_INDICATOR_TYPE is (BUTT, ROUND, PROJECTING SQUARE);
  LINE_CAP_INDICATOR_VALUE      :in LINE_CAP_INDICATOR_TYPE;

  procedure SET_LINE_CAP
    (LINE_CAP_INDICATOR_VALUE      :in LINE_CAP_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

Proposal Number: 38

Date of Presentation: 10 April 1987

Sponsoring Authority: ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Set Line Mitre Limit

#### Description

This escape function sets the line mitre limit value. This value helps determine the shape put at corners between portions of lines primitives. Its purpose is to place a limit on how long a "spike" can emanate from the join of two portions of a line primitive by "truncating" long mitre joins into bevel joins. See attached sheets for additional details.

#### Additional Comments

None.

#### Justification for Inclusion

User specified mitre limits are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing. This proposal accompanies proposal 39 for a Set Line Join escape function.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.



Description:

**Set Line Mitre Limit** sets the line mitre limit value used in interpretation of line primitives to *mitre length specifier*, which must be a number greater than or equal to 1. Mitre limit is a dimensionless number that controls the treatment of corners between portions of line primitives when mitre joins have been specified. A related registered escape, **Set Line Join**, controls the type of join that is selected. When portions connect at a sharp angle, a mitre join results in a spike that extends well beyond the connection point. The purpose of the mitre limit is to cut off such spikes when they become objectionably long. If the mitre length specifier is less than or equal to 1, the default value of 1 shall be used.

At any given corner, the *mitre length* is the distance from the point at which the inner edges of the line portions intersect to the point in which the outside edges of the portions intersect (i.e., the diagonal length of the mitre). This distance increases as the angle between the portions decreases. Whenever the ratio of the mitre length to the line width exceeds the line mitre limit parameter and is also greater than 1.415, a bevel is introduced at the join perpendicular to the angle bisector and at the mitre limit. (Note that this is not, in general equivalent to introducing a bevel join when this limit is reached. Introducing such a join causes discontinuous behaviour, where small changes in the angle between the segments results in radically different appearances.) Whenever the ratio of the line mitre length to the line width exceeds the line mitre limit parameter and is also less than or equal to 1.415 (that is, when the line mitre limit parameter is between 1 and 1.415 inclusive), a bevel join is implemented at the mitre limit.

The ratio of line mitre length to line width is directly related to the angle  $\phi$  between the segments by the formula:

$$\text{mitre length} / \text{line width} = 1 / \sin (\phi/2)$$

Examples of mitre length specifier values are: 1.415 cuts off mitres (converts them to bevels) at angles less than 90 degrees, 2.0 cuts off mitres at angles less than 60 degrees, and 10.0 cuts mitres off at angles less than 11 degrees. The default value of line mitre limit is 10. Setting the line mitre limit to 1 cuts off mitres at all angles so that bevels are always produced even when mitres are specified. The lengths in the above formula must be in the same units (WC, VDC, etc.) and cancel out to yield a dimensionless mitre length specifier value. This escape applies to line primitives.

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Set Line Mitre Limit escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):  
mitre length specifier(R)

Items for Data Record:

mitre length specifier

Data Record Description:

The parameter defines the line mitre length specifier value.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

### 3) GKS Functional Description (reference ISO 7942 GKS functional description)

This escape is applicable at level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
mitre length specifier		R
output data record:		
none		

#### Errors:

- 8 GKS not in proper state: GKS shall be either in one of the states GKOP, WSOP, WSAC, or SGOP

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrS" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrS is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrS(MITRE)

Input Parameters:

REAL MITRE                      mitre length

Output Parameters:

NONE



b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

INTEGER IL	0
INTEGER RL	1
REAL RA( 1 )	mitre length
INTEGER SL	0

The Unpack Data Record function is not required by this escape.

5) **Pascal language binding** (reference: ISO/IEC 8651-2 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```

GREScapeDataIn = Record
    CASE EscapeID : GTEscapeDataTag of
        1: (
            U0001 MitreLengthSpecifier: REAL );
    END;

GREScapeDataOut = RECORD
    CASE EscapeId : GTEscapeDataTag of
        1: ( ) ;
    END;
    (*Null Record *)

```



6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3: Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SET\_LINE\_MITRE\_LIMIT" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set line mitre limit.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package GKS
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
    MITRE_LENGTH_SPECIFIER    :in ESCAPE_FLOAT;

procedure SET_LINE_MITRE_LIMIT
    (MITRE_LIMIT              :in MITRE_LENGTH;

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

**Proposal Number:** 39

**Date of Presentation:** 10 April 1987

**Sponsoring Authority:** ANSI

**Class of Graphical Item:** ESCAPE

**Specific Escape Function Identifier:** Set Line Join

**Description**

This escape function sets a line join value. This value determines the shape put at corners between portions of line primitives. See attached sheet for additional details.

**Additional Comments**

None.

**Justification for Inclusion**

User specified line joins are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing.

**Relationship to Standards**

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

Description:

Set Line Join sets the current line join value to *line join indicator*. This establishes the shape to be put at the corners between portions (line segments) of line primitives. The following line join values are defined:

**mitre join** : the outer edge of the two portions are extended until they meet at a point. (Note that the mitre limit value may affect the appearance of these joins.)

**round join** : a circular arc with diameter equal to the line width is drawn around the vertex between the adjoining segments and is filled in, producing a rounded corner.

**bevel join** : the meeting portions are finished with butt end cap and the resulting triangular notch is filled in.

If the line join indicator is not one of the defined values, the default value, mitre join, shall be used.

Join styles are significant only at points where consecutive portions of a line primitives connect at an angle; portions that meet or intersect fortuitously receive no special treatment.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Set Line Join escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):  
line join indicator (E)

Items for Data Record:

line join indicator

The following values of line join indicator are defined:

- 1: mitre join
- 2: round join
- 3: bevel join

Data Record Description:

The parameter defines the line join indicator value.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.



### 3) GKS Functional Description (reference ISO 7942 GKS functional description)

This escape is applicable at level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
line join indicator	(MITRE, ROUND, BEVEL)	E
output data record:		
none		

#### Errors:

- 8     *GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP*

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrs(LNJOIN)

Input Parameters:

INTEGER LNJOIN                      line join indicator (MITRE, ROUND, BEVEL)

Output Parameters:

NONE

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

line join indicator	mitre,	round,	bevel
INTEGER	GMITRE,	GROUND,	GBEVEL
PARAMETER	(GMITRE=1,	GROUND=2,	GBEVEL=3)

## Set Line Join

b) The following parameters are proposed for use when accessing this escape through the GESC function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

```
INTEGER IL 1
INTEGER 1A( 1 ) line join indicator (MITRE, ROUND, BEVEL)
INTEGER RL 0
INTEGER SL 0
```

The Unpack Data Record function is not required by this escape.

## 5) Pascal language binding (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEscapeLineJoin = (Mitre, Round, Bevel);
```

```
GREScapeDataIn = RECORD
  CASE EscapeId : GTEscapeDataTag of
    1: (
      U0001 LineJoinIndicator : GEscapeLineJoin);
  END;
```

```
GREScapeDataOut = RECORD
  CASE EscapeId : GTEscapeDataTag of
    1: ( ) ; (*Null Record*)
  END;
```

6) **GKS Ada Language binding** (reference ISO/IEC 8651-3 GKS Language Bindings; Part 3: Ada)

Registered ESCAPE's are in a library package name GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SET\_LINE\_JOIN" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set line join
--Data type ESCAPE_ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPES is
    type LINE_JOIN_INDICATOR_TYPE is (MITRE, ROUND, BEVEL);
    LINE_JOIN_INDICATOR_VALUE : in LINE_JOIN_INDICATOR_TYPE;

procedure SET_LINE_JOIN
    (LINE_JOIN_INDICATOR_VALUE : in LINE_JOIN_INDICATOR_TYPE );

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

<b>Date of Presentation:</b> 10 April 1987
--

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> GDP
-------------------------------------

<b>GDP Identifier:</b> Poly Cubic Bezier Curve
--

Description
-------------

The point set used with this GDP is divided into disjoint sets consisting of four consecutive points each. A separate Bezier cubic curve is drawn using each consecutive set of four points. The curve starts at the first point and ends at the fourth point within each set. The second and third points in each set are used as control points. In the event that the points within consecutive sets do not "overlap" (for example, the fourth point of one set does not have the same value as the first point in the next set), the curve segments are not implicitly connected. A poly Bezier curve is a line primitive that takes line attributes. See the attached sheets for a detailed description.

Additional Comments
---------------------

None.

Justification for Inclusion
-----------------------------

Bezier curves are widely available in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing.

Relationship to Standards
---------------------------

- 1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.
- 2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).



**Description:**

**Poly Cubic Bezier Curve** adds a Bezier cubic curve between the first point, referred to here as  $(X_0, Y_0)$  and the fourth point  $(X_3, Y_3)$ , using  $(X_1, Y_1)$  and  $(X_2, Y_2)$  as control points, for each consecutive set of four points in the point list. These consecutive sets of points are obtained by dividing the point list into disjoint sets of four consecutive points each. In the event that points within the consecutive sets do not "overlap" (for example, if the fourth point of one set does not have the same value as the first point of the next set), the curve segments are not implicitly connected. If the number of points in the point list is not an even multiple of four, then only those point sets (if any) that contain four points shall be used. Any remaining points shall be ignored.

Each set of four points defines the shape of its curve geometrically. The curve starts at  $(X_0, Y_0)$ , it is tangent to the line from  $(X_0, Y_0)$  to  $(X_1, Y_1)$  at that point, and it leaves the point in that direction. The curve ends at  $(X_3, Y_3)$ , it is tangent to the line from  $(X_2, Y_2)$  to  $(X_0, Y_0)$  at that point, and it approaches the point from that direction. The lengths of the lines  $(X_0, Y_0)$  to  $(X_1, Y_1)$  and  $(X_2, Y_2)$  to  $(X_3, Y_3)$  represent in some sense the "velocity" of the path at the endpoints. The curve is always contained in the convex hull of the four points.

The mathematical foundation of a Bezier cubic curve is derived from a pair of parametric cubic equations:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + x_0$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + y_0$$

The cubic section produced by **Cubic Bezier Curve** is the path traced by  $x(t)$  and  $y(t)$  as  $t$  ranges from 0 to 1. The Bezier control points corresponding to this curve are:

$$x_1 = x_0 + c_x/3$$

$$y_1 = y_0 + c_y/3$$

$$x_2 = x_1 + (c_x + b_x)/3$$

$$y_2 = y_1 + (c_y + b_y)/3$$

$$x_3 = x_0 + c_x + b_x + a_x$$

$$y_3 = y_0 + c_y + b_y + a_y$$

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Bezier curve generalized drawing primitive parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP)  
data record(D)

Items for Data Record:

The data record is empty (that is, it is a null string.)

Data Record Description:

The data record is empty.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a and above. It will use the polyline attribute set. The control points are transformed to NDC and the Bezier curve through those points is then drawn. A functional description of its input parameters is:

Name	Coordinate System	Values	Data Type	Range
number of points		(4)	I	
M 4-tuples of Bezier control points	WC		Mx4xP	>0
GDP identifier N		as assigned		
GDP data record:		empty		

#### Errors:

- 5 *GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP*
- 100 *Number of points is invalid: the number of points shall be a multiple of four and shall be greater than zero*

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDPqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GDPqrs( N, PXA, PYA )
```

Input Parameters:

```
INTEGER N           number of points           (M*4)
REAL PXA(M*4), PYA(M*4) 4-tuples of Bezier control points
```

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

```
INTEGER IL 0
INTEGER RL 0
INTEGER SL 0
```

The Unpack Data Record function is not required by this escape.



5) **Pascal language binding** (reference: ISO/IEC 8651-2 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```

NumPoints = M*4;
Points : GAPointArray;

GRGDPData = RECORD
    CASE GDPId : GTGDPDataTag OF
        1: (); (* null data record*)
    END;

```

6) **GKS Ada language binding** (reference ISO/IEC 8651-3 GKS Language Bindings; Part 3: Ada)

Registered GDP's are in a library package named GKS\_GDP. GKS Ada provides a data type package, GKS\_TYPES which provides types declarations.

The binding for the "procedure BEZIER\_CURVE" form (as defined in subclause 4.1 of the GKS Ada language binding) of the GDP is:

```

--
--GDP function for a Bezier curve.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
    type BEZIER_POINTS is new WC.POINT_ARRAY (1...);

    procedure BEZIER_CURVE
        (BEZIER_CONTROL_POINTS      :in BEZIER_POINTS;
         GDP_IDENTIFIER              :in GDP_ID);

    --
    --more GDP procedures can be inserted here
    --

end GKS_GDP;

```



<b>Date of Presentation:</b> 10 April 1987
--

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> GDP
-------------------------------------

<b>GDP Identifier:</b> Conic Arc
----------------------------------

Description
-------------

A bounded, connected portion of a parent conic curve is defined in a definition space and then transformed into an appropriate standard-dependent coordinate system by the conic arc transformation matrix. The conic arc transformation matrix is defined by the registered escape function Set Conic Arc Transformation Matrix. A Conic Arc is a line primitive that takes line attributes. See the attached sheets for a detailed description.

Additional Comments
---------------------

None.

Justification for Inclusion
-----------------------------

Conic arcs are commonly found in proprietary graphics system. They are needed to support the requirements of office document and engineering drawing exchange.

Relationship to Standards
---------------------------

- 1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.
- 2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).

Description:

A conic arc is generated which is defined as follows:

A conic arc is a bounded connected portion of a parent conic curve which consists of more than one point. The parent arc is either an ellipse, a parabola, or a hyperbola. The conic arc is defined by a start point,  $P$ , an end point,  $Q$ , and six parameters,  $A, B, C, D, E$ , and  $F$ . The conic arc itself is defined by the six parameters and the following equation:

$$A \cdot X_t^2 + B \cdot X_t Y_t + C \cdot Y_t^2 + D \cdot X_t + E \cdot Y_t + F = 0$$

where  $(X_t, Y_t)$  is an abstract Cartesian coordinate system called "definition space". In order for the conic arc to be processed correctly by the receiving system given the above presentation, the conic arc entity must be positioned such that each of its axes is parallel to either the  $X_t$  axis or  $Y_t$  axis. The arc is then positioned correctly in the appropriate computer graphics coordinate system by using the value of the current Conic Arc Transformation Matrix.

To determine the form of the conic arc, the quantities  $Q1$ ,  $Q2$  and  $Q3$  are defined as follows:

$$Q1 = \text{determinant of } \begin{vmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{vmatrix}$$

$$Q2 = \text{determinant of } \begin{vmatrix} A & B/2 \\ B/2 & C \end{vmatrix}$$

$$Q3 = A + C$$

If  $Q2 > 0$  and  $(Q1 * Q3) < 0$ , then the arc is elliptical.

If  $Q2 < 0$  and  $Q1 < > 0$ , then the arc is hyperbolic.

If  $Q2 = 0$  and  $Q1 < > 0$ , the the arc is parabolic.

If  $Q1 = 0$  or if both  $Q2 > 0$  and  $(Q1 * Q3) > = 0$ , the results are implementation dependent.

In the case where the conic arc is elliptical, to distinguish the arc in question from its complement, the direction of the arc with respect to the definition space must be from start point to end point in a counter-clockwise direction.

In the case where the conic arc is parabolic or hyperbolic, the parameterization defines a unique portion of the parabola or a unique portion of a branch of the hyperbola, thus, the direction is irrelevant.

The direction of the conic arc with respect to the space where it is drawn is determined by the original direction of the arc in definition space, in conjunction with the action of the current Conic Arc Transformation Matrix.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

The start and end points are in definition space and are always encoded as real numbers. The arc is transformed to VDC using the current Conic Arc Transformation Matrix and is then drawn. A functional description of the conic arc parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP) - P, Q  
data record (D):

A  
B  
C  
D  
E  
F

Items for Data Record:

A  
B  
C  
D  
E  
F

Data Record Description:

The data record contains the six coefficients of the defining equation.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.



## Conic Arc

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a and above. It will use the polyline attribute set. The arc is transformed to NDC using the current Conic Arc Transformation Matrix and then drawn. A functional description of its input parameters is:

Name	Coordinate System	Values	Data Type
number of points		(2)	I
GPD identifier		as assigned	N
GDP data record (conic arc coefficients):			
A			R
B			R
C			R
D			R
E			R
F			R

#### Errors:

- 5 GKS not in proper state: GKS shall be in the state WSAC or in the state SGOP
- 100 Number of points is invalid: The number of points must be two (2).

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GDpqrs (P, Q, A, B, C, D, E, F)
```

Input Parameters:

```
REAL P(2), Q(2)      conic arc endpoints
REAL A, B, C, D, E, F  conic arc coefficients
```



b) The following parameters are proposed for use when accessing this GDP through the GGDP function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

```
INTEGER IL  0
INTEGER RL  6
REAL RA( 1 )  A
REAL RA( 2 )  B
REAL RA( 3 )  C
REAL RA( 4 )  D
REAL RA( 5 )  E
REAL RA( 6 )  F
INTEGER SL  0
```

The Unpack Data Record function is not required by this escape.

5) **Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
NumPoints = 2;

GRGDPData = RECORD
  CASE GDPIId : GTGDPDataTag OF
    1: (
      U0001 CoefA : REAL;
      U0001 CoefB : REAL;
      U0001 CoefC : REAL;
      U0001 CoefD : REAL;
      U0001 CoefE : REAL;
      U0001 CoefF : REAL);
  END;
```

6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

Registered GDP's are in a library package named GKS\_GDP. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure CONIC\_ARC" form (as defined in subclause 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a conic arc.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
    type CONIC_ARC_DATA_RECORD is new GDP_DATA_RECORD (0,6,0);

procedure CONIC_ARC
    (CONIC_ARC_RECORD          : in CONIC_ARC_DATA_RECORD);

--The components of the CONIC_ARC_DATA_RECORD are the start and end
--points as well as the coefficients specified in the registration
--procedure.
--
--more GDP procedures can be inserted here
--
end GKS_GDP;
```

<b>Date of Presentation:</b> 10 April 1987
--

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> ESCAPE
--

<b>Specific Escape Function Identifier:</b> Set Conic Arc Transformation Matrix
---

Description
-------------

This escape function sets a value of the transformation matrix needed to describe how a conic arc described by the registered GDP Conic Arc is moved from "definition space" to a standard-dependent graphic coordinate system. See attached sheets for a detailed description.

Additional Comments
---------------------

None.

Justification for Inclusion
-----------------------------

Conic arcs are commonly found in proprietary graphics system. They are needed to support the requirements of engineering drawing exchange. Due to various numerical problems, such curves are best specified in a "definition space" and then transformed to their final location by applying a transformation matrix. This escape function is needed to supply values for the required "modelling" or transformation matrix.

Relationship to Standards
---------------------------

- 1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.
- 2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a GDP. See attached sheets.



**Description:**

This escape is intended to work in conjunction with the conic arc generalized drawing primitive to transform the conic arc from definition space to "drawing space". Drawing space is a standard-dependent coordinate system, usually World Coordinates for API standards and Virtual Device Coordinates for metafile and device interface standards. The **conic arc transformation matrix** transforms definition space point coordinates by means of a matrix multiplication. This transformation is performed by applying the following matrix multiplication to coordinates:

$$\begin{array}{c} \begin{array}{ccc|c} R_{11} & R_{12} & R_{13} & 1 \\ R_{21} & R_{22} & R_{23} & 0 \\ \hline & & & 1 \end{array} \begin{array}{c} X_{in} \\ Y_{in} \\ X \\ Y_{in} \end{array} = \begin{array}{c} X_{out} \\ Y_{out} \\ X \\ Y_{out} \end{array} \end{array}$$

where  $|R_{ij}|$  is the transformation matrix,  $(X_{in}, Y_{in})$  is the coordinate to be transformed and  $(X_{out}, Y_{out})$  is the coordinate resulting from the transformation. Both the input and output coordinate systems are assumed to be orthogonal, Cartesian and right-handed. The default transformation matrix is:

$$\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ \hline & & & 1 \end{array}$$

[Note: IGES version 3.0 defines this transformation as a matrix multiplication followed by a vector addition (translation). To be consistent with the way transformations are defined in computer graphics standards, an equivalent homogeneous coordinate formulation is used here instead. To translate this form to the IGES form, simply use the first two columns of the above matrix as the IGES matrix and the last column as the "T-vector" values T1 and T2 in IGES.]

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM;  
Part 1: Functional Description)

This matrix is used to transform a Conic Arc element from its definition space into VDC where it is drawn. A functional description of the Set Conic Arc Transformation Matrix escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):

R<sub>11</sub>  
R<sub>12</sub>  
R<sub>13</sub>  
R<sub>21</sub>  
R<sub>22</sub>  
R<sub>23</sub>

Items for Data Record:

R<sub>11</sub>  
R<sub>12</sub>  
R<sub>13</sub>  
R<sub>21</sub>  
R<sub>22</sub>  
R<sub>23</sub>

Data Record Description:

The parameters define a 2x3 matrix used to transform a Conic Arc element from its definition space into VDC where it is drawn.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

### 3) GKS Functional Description (reference ISO 7942 GKS functional description)

This escape is applicable at level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
(transformation matrix)		2X3XR
R11		R
R12		R
R13		R
R21		R
R22		R
R23		R

output data record:  
none

#### Errors:

- 5 GKS not in proper state: GKS shall be in the state WSAC or in the state SGOP
- 100 Number of points is invalid

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrs(R11, R12, R13, R21, R22, R23)

#### Input Parameters:

REAL R11	2X3 transformation matrix
REAL R12	
REAL R13	
REAL R21	
REAL R22	
REAL R23	

#### Output Parameters:

NONE

## Set Conic Arc Transformation Matrix

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

```
INTEGER IL 0
INTEGER RL 6
REAL RA( 1 ) R11
REAL RA( 2 ) R12
REAL RA( 3 ) R13
REAL RA( 4 ) R21
REAL RA( 5 ) R22
REAL RA( 6 ) R23
INTEGER SL 0
```

The Unpack Data Record function is not required by this escape.

### 5) Pascal language binding (reference: ISO/IEC 8651-2 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GREscapeDataIn = RECORD
  CASE EscapeID : GTEscapeDataTag of
    1: (
      U0001 MatrixReal11 : REAL;
      U0001 MatrixReal12 : REAL;
      U0001 MatrixReal13 : REAL;
      U0001 MatrixReal21 : REAL;
      U0001 MatrixReal22 : REAL;
      U0001 MatrixReal23 : REAL);
  END;

GREscapeDataOut = RECORD
  CASE EscapeId : GTEscapeDataTag of
    1: ( ) ; (* Null Record*)
  END;
```



## Set Conic Arc Transformation Matrix

### 6) GKS Ada Language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SET\_CONIC\_ARC\_TRANSFORMATION\_MATRIX" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set conic arc transformation.
--Data type ESCAPE_ID and ESCAPE_FLOAT are defined in package
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is

    type CONIC_TRANSFORM_MATRIX is array (1..2,1..3) of ESCAPE_FLOAT;

    procedure SET_CONIC_ARC_TRANSFORMATION_MATRIX
        (CONIC_ARC_MATRIX      : in CONIC_TRANSFORM_MATRIX );

    --The components of CONIC_ARC_DATA are the 2X3 transformation
    --matrix specified in the registration proposal

    --
    --more ESCAPE procedures can be inserted here
    --
end GKS_ESCAPE;
```

**Proposal Number:** 43

**Date of Presentation:** 10 April 1987

**Sponsoring Authority:** ANSI

**Class of Graphical Item:** GDP

**GDP Identifier:** Parametric Spline Curve

**Description**

A planar (two dimensional) parametric spline curve is drawn. A parametric spline curve is a line primitive that takes line attributes. See attached sheets for a detailed description.

**Additional Comments**

None.

**Justification for Inclusion**

Parametric spline curves are commonly found in proprietary graphics systems. They are needed to support the requirements of engineering drawing exchange.

**Relationship to Standards**

- 1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.
- 2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).

Description:

A parametric spline curve as defined below is drawn. A parametric spline curve is a sequence of parametric polynomial segments. The curve is defined using the following parameters:

curve type (E)  
 N (number of segments) (I)  
 T (knot sequence for polynomial) ((N+1)R)  
 X coordinate polynomial list (N sets of four)

$A_X, B_X, C_X, D_X$  ((4\*N)R)

Y coordinate polynomial list (N sets of four)

$A_Y, B_Y, C_Y, D_Y$  ((4\*N)R).

This parametrization is generalized to allow for the representation of many different parametric spline curves using this one element. The curve type parameter indicates the type of parametric curve as it was represented in the sending system before being converted to this generic form. The following curve types are defined:

- 1) linear
- 2) quadratic
- 3) cubic
- 4) Wilson-Fowler
- 5) modified Wilson-Fowler
- 6) B spline

If the curve type is not one of the defined values, the default value, linear, shall be used.

The number of segments parameter, N, is the number of polynomial segments to be used to define the curve. Each segment is defined by a cubic polynomial in X and Y that is evaluated using the eight polynomial coefficients associated with that segment:  $A_X, B_X, C_X, D_X, A_Y, B_Y, C_Y, D_Y$ . Segment i is delimited by its knots,  $T(i)$  and  $T(i+1)$ . If the number of segments is less than one (1), the default value, one (1), shall be used.

The coordinates of the points of the i<sup>th</sup> segment of the curve are given by the following cubic polynomial equations. Note that coefficients D, or C and D will be zero if the polynomials are of degrees 2 or 1, respectively:

$$X(u) = A_X(i) + B_X(i)*s + C_X(i)*s^2 + D_X(i)*s^3$$

$$Y(u) = A_Y(i) + B_Y(i)*s + C_Y(i)*s^2 + D_Y(i)*s^3$$

where  $T(i) \leq u \leq T(i+1)$ ,  $i=1, \dots, N$  and  $s = u - T(i)$ . In order to avoid degeneracy, for each i at least one of the six real coefficients  $B_X, C_X, D_X, B_Y, C_Y, D_Y$  must be non-zero. If the knot sequence or the Y coordinate polynomial lists are invalid, the results are implementation dependent.

### Parametric Spline Curve

To enable determination of the terminate point and derivatives without computing the polynomials, the  $N^{\text{th}}$  polynomials and derivatives are evaluated at  $u = T(N+1)$ . These data, divided by the appropriate factorial (i.e. the second derivative divided by  $2!$ , the third by  $3!$ , etc.), are used as the  $N+1^{\text{st}}$  or terminate point values.



Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

The parameters of the curve are transformed to VDC and the curve is drawn. A functional description of the parametric spline curve parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

point list - empty  
data record (D): -

C (curve type) (Note 1) (E)  
N (number of segments) (I)  
T (knot sequence for polynomial) ((N+1)R)  
X coordinate polynomial list (N sets of four)

AX, BX, CX, DX ((4\*N)R)

Y coordinate polynomial list (N sets of four)

AY, BY, CY, DY ((4\*N)R).

**Note 1:** Curve type is one of (linear, quadratic, cubic, Wilson-Fowler, modified Wilson-Fowler, B spline)

Items for Data Record:

C  
N  
T(1)  
.....  
T(N+1)  
AX(1)  
BX(1)  
CX(1)  
DX(1)  
AX(2)  
BX(2)  
.....  
AY(1)  
BY(1)  
.....

Data Record Description:

The data record contains the parameters needed to define the parametric spline curve: curve type, number of segments, knot sequence list, and the coefficients of the two polynomials that define the X and Y coordinate values, respectively, for the curve.

## 2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

## 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a and above. It will use the polyline attribute set. The parameters of the curve are transformed to NDC and the curve is drawn. A functional description of its input parameters is:

Name	Coordinate System	Values	Data Type
number of points		(0)	I
GDP identifier		as assigned	N
GDP data record: see IGES attachments for definitions			
CTYPE		Note 1	I
NSEG			I
knot sequence for polynomials			(N+1) R
X coordinate polynomial list			((4*N) R)
Y coordinate polynomial list			((4*N) R) .

**Note 1:** Curve type is one of (linear, quadratic, cubic, Wilson-Fowler, modified Wilson-Fowler, B spline)

### Errors:

- 5 GKS not in proper state: GKS shall be in the state WSAC or in the state SGOP
- 100 Number of points is invalid

#### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GDpqrs( CTYPE, NSEG, T, X, Y )
```

Input Parameters:

INTEGER CTYPE	curve type
INTEGER NSEG	number of polynomial segments
REAL T(NSEG+1)	knot sequence list
REAL X(NSEG,4)	X coordinate polynomial coefficients
REAL Y(NSEG,4)	Y coordinate polynomial coefficients

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

curve type	linear, quadratic, cubic, Wilson-Fowler, modified Wilson-Fowler, B-spline
INTEGER	GLIN, GQUAD, GCUBIC, GWLFOW, GMWLFOW, GBSPLN
PARAMETER	(GLIN=1, GQUAD=2, GCUBIC=3, GWLFOW=4, GMWLFOW=5, GBSPLN=6)

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

```
Integer IL 2
Integer IA(1) C
Integer IA(2) N
Integer RL (9N+1) R
Real RA(1) T(1)
...
Real RA(N+1) T(N+1)
Real RA(N+2) AX(1)
Real RA(N+3) BX(1)
Real RA(N+4) CX(1)
Real RA(N+5) DX(1)
Real RA(N+6) AX(2)
Real RA(N+7) BX(2)
....
Real RA(5N+2) AY(1)
Real RA(5N+3) BY(1)
....
Integer SL 0
```

The Unpack Data Record function is not required by this escape.

5) **Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEGDPCurveProperty5 = (linear, quadratic, cubic,  
                        Wilson-Fowler, modified Wilson-Fowler,  
                        B-spline);
```

```
GRGDpdata = RECORD  
  CASE GDPid : GTGDpdataTag OF  
    1: (  
      U0001 CType : GEGDPCurveProperty5;  
      U0001 NSeg : INTEGER;  
      U0001 T : array [1..(NSeg+1)] of REAL;  
      U0001 X : array [1..(4*NSeg)] of REAL;  
      U0001 Y : array [1..(4*NSeg)] of REAL;  
    END;
```



6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part 3: Ada)

Registered GDP's are in a library package named GKS\_GDP. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure PARAMETRIC\_SPLINE\_CURVE" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a parametric spline curve gap.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
  type CURVE_TYPE is (linear, quadratic, cubic, Wilson-Fowler,
    modified Wilson-Fowler, B-spline);
  type KNOTS is array (SMALL NATURAL range<>) of
    ESCAPE_FLOAT;
  type SPLINE_CURVE_DATA_RECORD (NUM_SEG : SMALL_NATURAL := 0) is
    record
      SPLINE_TYPE      : CURVE_TYPE;
      KNOT_SEQUENCE    : KNOTS (1..(NUM_SEG+1));
      POLYNOMIALS      : POLYNOMIAL_ARRAY (1..NUM_SEG, 1..4);
    end record;

  procedure PARAMETRIC_SPLINE_CURVE
    (SPLINE_CURVE_RECORD      : in SPLINE_CURVE_DATA_RECORD);

  --The components of the SPLINE_CURVE_DATA_RECORD are those
  --specified in the registration proposal.
  --
  --more GDP procedures can be inserted here
  --
end GKS_GDP;
```

<b>Date of Presentation:</b> 10 April 1987
--

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> GDP
-------------------------------------

<b>GDP Identifier:</b> Rational B-Spline Curve
--

<b>Description</b>
--------------------

A planar (two dimensional) rational spline B-spline curve is drawn. A rational B-spline curve is a line primitive that takes line attributes. See attached sheets for a detailed description.

<b>Additional Comments</b>
----------------------------

None.

<b>Justification for Inclusion</b>
------------------------------------

Rational B-spline curves are commonly found in proprietary graphics system. They are needed to support the requirements of engineering drawing exchange.

<b>Relationship to Standards</b>
----------------------------------

- 1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.
- 2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).

Description:

A rational B-spline curve as defined below is drawn. The curve is defined using the following parameters:

$K$  (upper index of summation) (I)  
 $M$  (degree of the basis functions) (I)  
 curve open (one of: open, closed) (E)  
 equation type (one of: rational, polynomial) (E)  
 $T$  (knot sequence)  $((K+M+1)R)$   
 $W$  (weights)  $((K+1)R)$   
 $P$  (control points)  $((K+1)P)$   
 start param, end param (2R)

The parametric equation governing the definition of the rational B-spline curve is shown in the following expression:

$$G(t) = \frac{\sum_{i=0}^K W(i)P(i)b_i(t)}{\sum_{i=0}^K W(i)b_i(t)}$$

where  $W(i)$  are the weights,  $P(i)$  are the control points and  $b_i$  are the basis functions.

The B-spline basis functions,  $b_i$ , are non-negative piecewise polynomials of degree  $M$ .  $T$  is a non-decreasing sequence of real numbers  $T(-M), \dots, T(0), \dots, T(K+1)$ . Each function  $b_i$  is supported by the knot sequence interval  $[T(i-M), T(i+1)]$ . Between any two adjacent knot values,  $T(j)$  and  $T(j+1)$ , the corresponding basis function can be expressed as a single polynomial of degree  $M$ .

The curve itself is parametrized, with:

start param  $\leq t \leq$  end param,  
 $T(0) < \text{start param} < \text{end param} \leq T(K+1)$

Thus, for any parameter value  $t$  between  $T(0)$  and  $T(K+1)$ , the sum of the basis functions satisfies the following identity:

$$\sum_{i=0}^K b_i(t) = 1.$$

The B-Spline basis functions themselves are defined as follows. Let

$$N(t | t_{i-m}, \dots, t_{i+1})$$

denote the B-spline basis function of degree  $m$  supported on the interval  $[t_{i-m}, t_{i+1}]$ .

## Rational B-spline

The degree  $k$  functions are defined in terms of those of degree  $k-1$  as follows:

$$N(t | s_0, \dots, s_k) = \frac{(t - s_0)N(t | s_0, \dots, s_{k-1})}{s_{k-1} - s_0} + \frac{(s_k - t)N(t | s_0, \dots, s_k)}{s_k - s_1}$$

Since some denominators will be 0 in the case of multiple knots, the convention  $0/0 = 0$  is adopted in the above definition.

If the knot sequence, weights, or control points are invalid (i.e. they are not consistent or do not match the equation type) the results are implementation dependent. If *start param* is greater than *end param*, no curve shall be drawn. If the equation

If the beginning and ending points of the curve are identical, then *curve open* is set to *closed*, otherwise, it is set to *open*. If *curve open* is not one of the defined values, then the default value, *open*, shall be used. When *curve open* is *closed*, and the derived curve end points are not equal, the curve end points shall be forced to be equal by an implementation dependent method.

If all of the weights (i.e. elements of the weights array,  $W$ ) are not equal, then *equation type* is set to *rational*. Otherwise, if all of the weights are equal, then the weights cancel, the denominators sum to one and the equation type becomes *polynomial*. If *equation type* is not one of the defined values, then the default value, *rational*, shall be used. If the value of *equation type* is not consistent with the weight values, then the type equation implied by the weight values shall be substituted.



Relationship to particular standard:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

The parameters of the curve are transformed to VDC and the curve is then drawn. A functional description of the rational B-spline parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP) - contains the control points

data record (D):

K (upper index of sum) (I)  
M (degree of the basic functions) (I)  
curve open (one of: open, closed) (E)  
equation type (one of: rational, polynomial) (E)  
T (knot sequence) ((K+M+1)R)  
W (weights) ((K+1)R)  
P (control points) ((K+1)P)  
start param, end param (2R)

Items for Data Record:

K  
M  
curve open  
equation type  
T(-M)  
T(-M+1)  
....  
T(K+1)  
W(0)  
....  
W(K+1)  
start param  
end param

Data Record Description:

The data record contains the parameters that define the rational B-spline curve: the upper index of summation, the basis functions, curve open, equation type, the knot sequence, the weights, and two parameters -- start param and end param -- that determine the beginning and end of the curve.

## 2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

## 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a and above. It will use the polyline attribute set. The control points are transformed to NDC and the curve is drawn. A functional description of its input parameters is:

Name	Coordinate System	Values	Data Type
number of points			I
control points	WC		nxP
GDP identifier		as assigned	N
GDP data record:			
K (upper index of summation)			I
M (degree of basis functions)			I
CUROPN		(open,closed)	E
EQNTYP		(rational,polynomial)	E
T			R
W			R
SPARAM, EPARAM			R

### Errors:

- 5 GKS not in proper state: GKS shall be in the state WSAC or in the state SGOP
- 100 Number of points is invalid

4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GDpqrs(N, PXA, PYA, K, M, CUROPN, EQNTYP, T,
+                W, SPARAM, EPARAM)
```

Input Parameters:

INTEGER N	number of control points
REAL PXA(*), PYA(*)	control points
INTEGER K	upper index of summation
INTEGER M	degree of basis functions
INTEGER CUROPN	
INTEGER EQNTYP	
REAL T( N+2M )	knot sequence
REAL W( K )	weight
REAL SPARAM, EPARAM	determines start and end points

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

curve open	open,	closed
INTEGER	GCROPN,	GCRCLCLO
PARAMETER	(GCROPN=0,	GGCRCLCLO=1)
equation type	rational,	polynomial
INTEGER	GEQRAT,	GEQPOL
PARAMETER	(GEQRAT =0,	GEQRAT =1)

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

```

Integer IL 4
Integer IA(1) K
Integer IA(2) M
Integer IA(3) CUROPN
Integer IA(4) EQNTYP
Integer RL 2K + M + 4
Real RA(1) T(-M)
Real RA(2) T(-M+1)
....
Real RA(K+M+1) T(K+1)
Real RA(K+M+2) W(0)
....
Real RA(2K+M+2) W(K+1)
Real RA(2K+M+3) SPARAM
Real RA(2K+M+4) EPARAM
Integer SL 0

```

The Unpack Data Record function is not required by this escape.

#### 5) Pascal language binding (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```

GEGDPCurveProperty2 = (GVGDPOpen, GVGDPClosed);
GEGDPCurveProperty3 = (GVGDPRational, GVGDPPolynomial);

Points          : GAPointArray;      (*control points*)

GRGDpData = RECORD
  CASE GDPId : GTGDpDataTag OF
    1: (
      U0001 K          : INTEGER;
      U0001 M          : INTEGER;
      U0001 CurveOpen  : GEGDPCurveProperty2;
      U0001 EquationType : GEGDPCurveProperty3;
      U0001 T          : REAL;
      U0001 W          : REAL;
      U0001 StartParam  : REAL;
      U0001 EndParam    : REAL;
    )
  END;

```



6) **GKS Ada language binding** (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3: Ada)

a) Registered GDP's are in a library package named GKS\_GDP. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure Rational\_B\_SPLINE\_CURVE" form (as defined in subclause 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a rational B_Spline curve.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS-TYPES;
use GKS-TYPES;
package GKS_GDP is
  type CURVE_OPEN_TYPE is (OPEN, CLOSED);
  CURVE_OPEN_VALUE : in CURVE_OPEN_TYPE;

  type EQUATION_TYPE is (RATIONAL, POLYNOMIAL);
  EQUATION_VALUE : in EQUATION_TYPE;

  type KNOT_SEQUENCE_ARRAY is array (SMALL NATURAL range<>) of
    ESCAPE_FLOAT;

  type WEIGHT_SEQUENCE_ARRAY is array (SMALL NATURAL range<>) of
    ESCAPE_FLOAT;

  type RATIONAL_B_SPLINE_DATA_RECORD is
    record
      UPPER_INDEX_OF_SUM           : INTEGER;
      DEGREE_OF_BASIS_FUNCTIONS    : INTEGER;
      CURVE_OPEN_VALUE             : CURVE_OPEN_TYPE;
      EQUATION_VALUE               : EQUATION_TYPE;
      KNOT_SEQUENCE                : KNOT_SEQUENCE_ARRAY;
      WEIGHT_SEQUENCE              : WEIGHT_SEQUENCE_ARRAY;
      START_PARAM                 : ESCAPE_FLOAT;
      END_PARAM                   : ESCAPE_FLOAT;
    end record;

  procedure RATIONAL_B_SPLINE_CURVE
    (CONTROL_POINTS :in WC.POINT_ARRAY;
     RATIONAL_B_SPLINE_DATA_RECORD :in GDP_DATA_RECORD);

--The components of the RATIONAL_B_SPLINE_DATA_RECORD are those
--specified in the registration proposal..
--
--more GDP procedures can be inserted here
--
end GKS_GDP;
```

PART 2

REVISED PROPOSALS FROM LETTER BALLOT #76.

TEXT READY FOR REVIEW AND THEN RE-BALLOTING.



**Proposal Number:** 45

**Date of Presentation:** 10 April 1987

**Sponsoring Authority:** ANSI

**Class of Graphical Item:** ESCAPE

**Specific Escape Function Identifier:** Set Edge Mitre Limit

**Description**

This escape function sets a value for the current edge mitre limit. This value determines the shape put at corners between consecutive edges of filled area graphical primitives. Its purpose is to place a limit on how long a "spike" can emanate from the join of two portions of an edge of a filled area primitive by "truncating" long mitre joins into bevel joins. See attached sheets for additional details.

**Additional Comments**

None.

**Justification for Inclusion**

User specified mitre limits are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing.

**Relationship to Standards**

- 1) ISO 7942 (GKS) - Not applicable.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Not applicable.



Description:

**Set Edge Mitre Limit** sets the edge mitre limit value used in interpretation of filled area primitives to *mitre length specifier*, which must be a number greater than or equal to 1. Mitre limit is a dimensionless number that controls the treatment of corners between portions of edges of filled area primitives when mitre joins have been specified. A related registered escape, **Set Edge Join**, controls the type of join that is selected. When portions connect at a sharp angle, a mitre join results in a spike that extends well beyond the connection point. The purpose of the mitre limit is to cut off such spikes when they become objectionably long. If the mitre length specifier is less than or equal to 1, the default value of 1 shall be used.

At any given corner, the *mitre length* is the distance from the point at which the inner edges intersect to the point in which the outside edges intersect (i.e., the diagonal length of the mitre). This distance increases as the angle between the edges decreases. Whenever the ratio of the mitre length to the line width exceeds the edge mitre limit parameter and is also greater than 1.415, a bevel is introduced at the join perpendicular to the angle bisector and at the mitre limit. (Note that this is not, in general equivalent to introducing a bevel join when this limit is reached. Introducing such a join causes discontinuous behaviour, where small changes in the angle between the segments results in radically different appearances.) Whenever the ratio of the edge mitre length to the edge width exceeds the edge mitre limit parameter and is also less than or equal to 1.415 (that is, when the edge mitre limit parameter is between 1 and 1.415 inclusive), a bevel join is implemented at the mitre limit.

The ratio of edge mitre length to edge width is directly related to the angle  $\phi$  between the segments by the formula:

$$\text{mitre length} / \text{line edge} = 1 / \sin (\phi/2)$$

Examples of mitre length specifier values are: 1.415 cuts off mitres (converts them to bevels) at angles less than 90 degrees, 2.0 cuts off mitres at angles less than 60 degrees, and 10.0 cuts mitres off at angles less than 11 degrees. The default value of edge mitre limit is 10. Setting the edge mitre limit to 1 cuts off mitres at all angles so that bevels are always produced even when mitres are specified. The lengths in the above formula must be in the same units (WC, VDC, etc.) and cancel out to yield a dimensionless mitre length specifier value. This escape applies to filled area primitives.

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Edge Mitre Limit escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):  
mitre length specifier(R)

Items for Data Record:

mitre length specifier

Data Record Description:

The parameter defines the mitre length specifier.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) **GKS Functional Specification** (reference ISO 7492, GKS Functional Specification)

This escape does not apply to GKS since GKS does not have separate edge attributes.

4) **GKS FORTRAN language binding** (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

This escape does not apply to GKS since GKS does not have separate edge attributes.

5) **GKS Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

This escape does not apply to GKS since GKS does not have separate edge attributes.

6) **GKS Ada language binding** (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

This escape does not apply to GKS since GKS does not have separate edge attributes.



Date of Presentation: 10 April 1987

Sponsoring Authority: ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Set Edge Cap

#### Description

This escape function sets a value for the current edge cap. This value is used to determine the shape put at the corners where consecutive edges of filled area graphical primitives meet. See attached sheet for additional details.

#### Additional Comments

None.

#### Justification for Inclusion

User specified edge caps are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Not applicable.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Not applicable.



Description:

Set Edge Cap selects the edge cap value specified by the edge cap indicator. This value determines the shape to be put at "corners" of the edge of filled area primitives. The shape is not applied to the ends of individual "dashes" that compose some edgetypes. The following edge cap values are defined with this proposal:

**butt cap** : the edge is squared off at the endpoint; there is no projection beyond the endpoint.

**round cap** : a semicircular arc with diameter equal to the edge width is drawn around the corner point and filled. The drawn line thus projects beyond the corner point.

**projecting square cap** : the edge is squared off at a distance equal to half the edge width beyond the corner point.

If the edge cap value is not one of the defined values, the default value, butt cap, shall be used.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Edge Cap escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):  
edge cap indicator (E)

Items for Data Record:

edge cap indicator

The following edge cap indicator values are defined:

- 1: butt cap
- 2: round cap
- 3: projecting square cap

Data Record Description:

The parameter defines the edge cap indicator.

**2) CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

**3) GKS Functional Specification** (reference ISO 7492, GKS Functional Specification)

This escape does not apply to GKS since GKS does not have separate edge attributes.

**4) GKS FORTRAN language binding** (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

This escape does not apply to GKS since GKS does not have separate edge attributes.

**5) GKS Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

This escape does not apply to GKS since GKS does not have separate edge attributes.

**6) GKS Ada language binding** (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

This escape does not apply to GKS since GKS does not have separate edge attributes.

<b>Date of Presentation:</b> 10 April 1987
--

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> ESCAPE
--

<b>Specific Escape Function Identifier:</b> Set Edge Join
---

Description
-------------

This escape function sets a value for the current edge join. This value determines the shape put at corners between consecutive edges of filled area graphical primitives. See attached sheet for additional details.

Additional Comments
---------------------

None.

Justification for Inclusion
-----------------------------

User specified edge joins are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing.

Relationship to Standards
---------------------------

- 1) ISO 7942 (GKS) - Not applicable.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Not applicable.

**Description:**

**Set Edge Join** sets the current edge join value to *edge join indicator*. This establishes the shape to be put at the corners between portions of edges of filled area primitives. The following edge join values are defined:

**mitre join** : the outer edge of the two portions are extended until they meet at a point. (Note that the mitre limit value may affect the appearance of these joins.)

**round join** : a circular arc with diameter equal to the edge width is drawn around the vertex between the adjoining segments and is filled in, producing a rounded corner.

**bevel join** : the meeting portions are finished with butt end cap and the resulting triangular notch is filled in.

If the edge join indicator is not one of the defined values, the default value, mitre join, shall be used.

Join styles are significant only at points where consecutive edges of filled area primitives connect at an angle; portions that meet or intersect fortuitously receive no special treatment.



Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Edge Join escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):  
edge join indicator (E)

Items for Data Record:

edge join indicator

The following values of edge join indicator are defined:

1: mitre join  
2: round join  
3: bevel join

Data Record Description:

The parameter defines the edge join indicator value.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) **GKS Functional Specification** (reference ISO 7492, GKS Functional Specification)

This escape does not apply to GKS since GKS does not have separate edge attributes.

4) **GKS FORTRAN language binding** (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

This escape does not apply to GKS since GKS does not have separate edge attributes.

Set Edge Join

5) **GKS Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

This escape does not apply to GKS since GKS does not have separate edge attributes.

6) **GKS Ada language binding** (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

This escape does not apply to GKS since GKS does not have separate edge attributes.

<b>Date of Presentation:</b> 18 July 1988
---

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> ESCAPE
--

<b>Specific Escape</b>	<b>Function Identifier:</b> Select Typeface Posture
------------------------	---

#### Description

Select Typeface Posture selects a desired typeface posture. This information is used to indicate a preference for one posture over others when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. See attached sheet for additional details.

#### Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DIS 9541).

#### Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable in office systems and graphic arts applications. Most proprietary graphics systems currently use a typographic text model that includes a posture attribute. This escape allows the posture of a typeface to be selected for font substitution or selection purposes.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

**Description:**

**Select Typeface Posture** sets the desired typeface posture to the value specified by the *typeface posture indicator*. This information is used to indicate a preference for one posture over others when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. The following posture values (from ISO/DIS 9541) are defined:

- upright
- oblique
- back slanted oblique
- italic
- back slanted italic

If the *typeface posture indicator* is not one of the defined values, then the default value, upright, shall be used.



Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Select Typeface Posture escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):

typeface posture indicator (upright, oblique, back slanted  
oblique, italic, back slanted italic) (E)

Items for Data Record:

typeface posture indicator

The following typeface posture indicator values are defined:

- 1: upright
- 2: oblique
- 3: back slanted oblique
- 4: italic
- 5: back slanted italic

Data Record Description:

The parameter defines the desired typeface posture.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N

input data record:

typeface posture indicator	Note 1	E
----------------------------	--------	---

Note 1. Typeface posture indicator is one of: (upright, oblique, back slanted oblique, italic, back slanted italic)

output data record:

none

#### Errors:

8 GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrs (TFPOS)

Input Parameters:

INTEGER TFPOS	typeface posture indicator (upright, oblique, back slanted oblique, italic, back slanted italic)
---------------	--

Output Parameters:

NONE

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

typeface posture	upright, italic,	oblique, back slanted italic	back slanted oblique,
INTEGER	GTFUP, GTFITL,	GTFOBL, GTFBSI	GTFBSO,
PARAMETER	(GTFUP =1, GTFITL=4,	GTFOBL =2, GTFBSI=5)	GTFBSO=3,

## Select Typeface Posture

b) The following parameters are proposed for use when accessing this escape through the GESC function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

```
INTEGER IL          1
INTEGER IA( 1 )     posture indicator
INTEGER RL 0
INTEGER SL 0
```

The Unpack Data Record function is not required by this escape.

**5) Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEScapeTypefacePostureType = (GVEscapeUpright, GVEscapeOblique,
                               GVEscapeBackSlantedOblique,
                               GVEscapeItalic,
                               GVEscapeBackSlantedItalic);
```

```
GREScapeDataIn = RECORD
```

```
    CASE EscapeID : GTEscapeDataTag of
        1: (
            TextPosture :GEScapeTypefacePostureType);
    End;
```

```
GREScapeDataOut = RECORD
```

```
    CASE EscapeId : GTEscapeDataTag of
        1: () ; (*Null Record *)
    END;
```



## Select Typeface Posture

6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SELECT\_TYPEFACE\_POSTURE" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for typeface posture.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type TYPEFACE_POSTURE_INDICATOR_TYPE is (UPRIGHT, OBLIQUE,
    BACK_SLANTED_OBLIQUE, ITALIC, BACK_SLANTED_ITALIC);
  type TYPEFACE_POSTURE_DATA_RECORD is
    TYPEFACE_POSTURE_INDICATOR_VALUE      : in
      TYPEFACE_POSTURE_INDICATOR_TYPE;

  procedure SELECT_TYPEFACE_POSTURE

    (TYPEFACE_POSTURE_INDICATOR_VALUE : in
      TYPEFACE_POSTURE_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

Date of Presentation:	18 July 1988
-----------------------	--------------

Sponsoring Authority:	ANSI
-----------------------	------

Class of Graphical Item:	ESCAPE
--------------------------	--------

Specific Escape	Function Identifier:	Select Typeface Structure
-----------------	----------------------	---------------------------

#### Description

Select Typeface Structure selects a desired typeface structure. This information is used to indicate a preference for one structure over others when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. See attached sheet for additional details.

#### Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

#### Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable in office systems and graphic arts applications. Many proprietary graphics systems currently use a typographic text model that includes a structure attribute. This escape allows the structure of the strokes of the shapes of the glyphs in a typeface to be selected for font substitution or selection purposes.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

Description:

**Select Typeface Structure** sets the desired typeface structure to the value specified by the typeface structure indicator. This information is used to indicate a preference for one structure over others when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. The following structure values (from ISO/DIS 9541) are defined:

- solid
- outline
- inline
- shadow
- patterned

If the typeface structure indicator is not one of the defined values, then the default value, solid, shall be used.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Select Typeface Structure escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):  
typeface structure indicator (solid, outline, inline,  
shadow, patterned) (E)

Items for Data Record:

typeface structure indicator

The following typeface structure indicator values are defined:

- 1: solid
- 2: outline
- 3: inline
- 4: shadow
- 5: patterned

Data Record Description:

The parameter defines the desired typeface structure.



**2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)**

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

## Select Typeface Structure

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N

input data record:

typeface structure indicator	Note 1	E
------------------------------	--------	---

**Note 1.** Typeface structure indicator is one of: (solid, outline, inline, shadow, patterned)

output data record:

none

#### Errors:

- 8 GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP

#### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrs (TFSTRC)

Input Parameters:

INTEGER TFSTRC                      typeface structure indicator (solid,  
outline, inline, shadow, patterned)

Output Parameters:

NONE

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

typeface structure	solid, shadow,	outline, patterned	inline,
INTEGER	GTFSOL, GTFSHD,	GTFOTL, GTFPAT	GTFINL,
PARAMETER	(GTFSOL =1, GTFOOTL =2, GTFINL=3, GTFSHD=4, GTFPAT=5)		

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

INTEGER IL	1
INTEGER IA (1)	typeface structure indicator
INTEGER RL 0	
INTEGER SL 0	

The Unpack Data Record function is not required by this escape.

## Select Typeface Structure

### 5) Pascal language binding (reference: ISO/IEC 8651-2 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEScapeTypefaceStructureType =  
    (GVEscapeSolid, GVEscapeOutline,  
     GVEscapeInline, GVEscapeShadow,  
     GVEscapePatterned);
```

```
GREScapeDataIn = RECORD
```

```
    CASE EscapeID : GTEscapeDataTag of  
        1: (  
            U0001 TypefaceWeight  
                :GEScapeTypefaceStructureType);  
    End;
```

```
GREScapeDataOut = RECORD
```

```
    CASE EscapeId : GTEscapeDataTag of  
        1: () ; (*Null Record *)  
    END;
```



6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SELECT\_TYPEFACE\_STRUCTURE" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for typeface structure.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type TYPEFACE_STRUCTURE_INDICATOR_TYPE is (SOLID, OUTLINE,
    INLINE, SHADOW, PATTERNED);
  type TYPEFACE_STRUCTURE_DATA_RECORD is
    TYPEFACE_STRUCTURE_INDICATOR_VALUE      :in
    TYPEFACE_STRUCTURE_INDICATOR_TYPE;

  procedure SELECT_TYPEFACE_STRUCTURE

    (TYPEFACE_STRUCTURE_INDICATOR_VALUE : in
      TYPEFACE_STRUCTURE_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

Date of Presentation: 18 July 1988
------------------------------------

Sponsoring Authority: ANSI
----------------------------

Class of Graphical Item: ESCAPE
---------------------------------

Specific Escape	Function Identifier: Select Typeface Scores
-----------------	---

#### Description

Select Typeface Scores selects the desired typeface scores. It allows selection of right scores (underscore in left-to-right writing mode (DIS 9541) or RIGHT Text Path (computer graphics standards)), left scores (overscore in left-to-right writing mode (DIS 9541) or RIGHT Text Path (computer graphics standards)), and through scores (scores drawn through the centre of glyphs) This information is used to indicate a preference for the [resence of scores when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. See attached sheets for additional details.

#### Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DIS 9541).

#### Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable in office systems and graphic arts applications. Most proprietary graphics systems currently use a typographic text model that allows scoring as one attribute. Examples of such scores are underscores (or underline), through scores, and overscores. This escape allows presence of scores in a typeface to be selected for font substitution or selection purposes.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

Description:

**Select Typeface Scores** selects one of more typeface scores as specified by the values of *right score indicator*, *left score indicator*, and *through score indicator*. This information is used to indicate a preference for the presence of scores when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. The following score indicators (from ISO/DIS 9541) are supported:

- right score indicator (an underscore in left-to-right writing mode)
- left score indicator (an overscore in left-to right writing mode)
- through score indicator (a score located through the "centre" of the glyphs)

The allowable values for each score indicator are:

- off : the score is absent
- on : the score is present

If any *score indicator* is not one of the defined values, then the default value, off, shall be used.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Select Typeface Scores escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):

right score indicator (off, on) (E)  
left score indicator (off, on) (E)  
through score indicator (off, on) (E)

Items for Data Record:

right score indicator  
left score indicator  
through score indicator

The following values for each score indicator are defined:

0: off  
1: on

Data Record Description:

The parameter defines the desired typeface scores.



**2) CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N

input data record:

right score indicator	(off, on)	E
left score indicator	(off, on)	E
through score indicator	(off, on)	E

output data record:  
none

#### Errors:

8 GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrS" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrS is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrS (TFSCOR)

Input Parameters:

INTEGER TFRSCR	right score indicator (off, on)
INTEGER TFLSCR	left score indicator (off, on)
INTEGER TFTSCR	through score indicator (off, on)

Output Parameters:

NONE

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

typeface score indicator	on,	off
INTEGER	GTFSON,	GTFSOFF
PARAMETER	(GTFSON=0,	GTFSOFF=1)

· Select Typeface Scores

b) The following parameters are proposed for use when accessing this escape through the GESC function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

INTEGER IL	3
INTEGER IA (1)	right score indicator (off, on)
INTEGER IA (2)	left score indicator (off, on)
INTEGER IA (3)	through score indicator (off, on)
INTEGER RL 0	
INTEGER SL 0	

The Unpack Data Record function is not required by this escape.

## Select Typeface Scores

### 5) Pascal language binding (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEscapeTypefaceScoreType =      (GVEscapeScoreOff,  
                                  GVEscapeScoreOn);
```

```
GREscapeDataIn = RECORD
```

```
    CASE EscapeID : GTEscapeDataTag of  
      1: (  
        U0001 RightScoreIndicator  
          :GEscapeTypefaceScoreType);  
        U0001 LeftScoreIndicator  
          :GEscapeTypefaceScoreType);  
        U0001 ThroughScoreIndicator  
          :GEscapeTypefaceScoreType);  
    End;
```

```
GREscapeDataOut = RECORD
```

```
    CASE EscapeId : GTEscapeDataTag of  
      1: () ; (*Null Record *)  
    END;
```



6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SELECT\_TYPEFACE\_SCORES" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for typeface scores.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type TYPEFACE_SCORE_INDICATOR_TYPE is (OFF, ON);
  type TYPEFACE_SCORE_DATA_RECORD is
    record
      RIGHT_SCORE_INDICATOR_VALUE      : in
                                         TYPEFACE_SCORE_INDICATOR_TYPE;
      LEFT_SCORE_INDICATOR_VALUE       : in
                                         TYPEFACE_SCORE_INDICATOR_TYPE;
      THROUGH_SCORE_INDICATOR_VALUE    : in
                                         TYPEFACE_SCORE_INDICATOR_TYPE;
    end record;

  procedure SELECT_TYPEFACE_SCORES
    (TYPEFACE_SCORE_RECORD : in TYPEFACE_SCORE_DATA_RECORD);

  --
  --more ESCAPE procedures can be inserted here
  --
end GKS_ESCAPE;
```

Date of Presentation: 18 July 1988

Sponsoring Authority: ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Select Typeface Weight

#### Description

Select Typeface Weight selects a desired typeface weight. This information is used to indicate a preference for one weight over others when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. See attached sheet for additional details.

#### Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

#### Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable in office systems and graphic arts applications. Most proprietary graphics systems currently use a typographic text model that includes a weight attribute. This escape allows the weight of a typeface to be selected for font substitution or selection purposes.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

Description:

**Select Typeface Weight** sets the desired typeface weight to the value specified by the *typeface weight indicator*. This information is used to indicate a preference for one weight over others when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. The following weight values (from ISO/DIS 9541) are defined:

- ultra light
- extra light
- light
- semi light
- medium
- semi bold
- bold
- extra bold
- ultra bold

If the *typeface weight indicator* is not one of the defined values, then the default value, medium, shall be used.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Select Typeface Posture escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):

typeface weight indicator (ultra light, extra light,  
light, semi light, medium, semi bold, bold, extra bold,  
ultra bold) (E)

Items for Data Record:

typeface weight indicator

The following typeface weight indicator values are defined:

- 1: ultra light
- 2: extra light
- 3: light
- 4: semi light
- 5: medium
- 6: semi bold
- 7: bold
- 8: extra bold
- 9: ultra bold

Data Record Description:

The parameter defines the desired typeface weight.



2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

Select Typeface Weight

3) **GKS Functional Specification** (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
typeface weight indicator	Note 1	E

**Note 1.** Typeface weight indicator is one of: (ultra light, extra light, light, semi light, medium, semi bold, bold, extra bold, ultra bold)

output data record:  
none

Errors:

8 *GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP*

4) **GKS FORTRAN language binding** (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrS" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrS is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrS(TFWGHT)

Input Parameters:

INTEGER TFWGHT                      typeface weight indicator (ultra light, extra light, light, semi light, medium, semi bold, bold, extra bold, ultra bold)

Output Parameters:

NONE

## Select Typeface Weight

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

typeface weight	ultra light, light, medium, bold, ultra bold	extra light, semi light, semi bold, extra bold,
INTEGER	GTFUL, GTFLIT, GTFMED, GTFBLD, GTFUB	GTFEL, GTFSL, GTFEB, GTFEB,
PARAMETER	(GTFUL=1, GTFLIT=3, GTFMED=5, GTFBLD=7, GTFUB=9)	GTFEL=2, GTFSL=4, GTFEB=6, GTFEB=8,

b) The following parameters are proposed for use when accessing this escape through the GESC function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

INTEGER IL	1
INTEGER IA (1)	typeface weight indicator
INTEGER RL 0	
INTEGER SL 0	

The Unpack Data Record function is not required by this escape.

**5) Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```

GEScapeTypefaceWeightType =      (GVEscapeUltraLight,
                                   GVEscapeExtraLight,
                                   GVEscapeLight,
                                   GVEscapeSemiLight,
                                   GVEscapeMedium,
                                   GVEscapeSemiBold,
                                   GVEscapeBold,
                                   GVEscapeExtraBold,
                                   GVEscapeUltraBold);

```

```

GREScapeDataIn = RECORD
    CASE EscapeID : GTEscapeDataTag of
        1: (
            U0001 TypefaceWeight
              :GEScapeTypefaceWeightType);
    End;

```

```

GREScapeDataOut = RECORD
    CASE EscapeId : GTEscapeDataTag of
        1: ( ) ; (*Null Record *)
    END;

```



6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SELECT\_TYPEFACE\_WEIGHT" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for typeface weight.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type TYPEFACE_WEIGHT_INDICATOR_TYPE is (ULTRA_LIGHT, EXTRA_LIGHT,
    LIGHT, SEMI_LIGHT, MEDIUM, SEMI_BOLD, BOLD, EXTRA_BOLD,
    ULTRA_BOLD);
  type TYPEFACE_WEIGHT_DATA_RECORD is
    TYPEFACE_WEIGHT_INDICATOR_VALUE      : in
                                          TYPEFACE_WEIGHT_INDICATOR_TYPE;

  procedure SELECT_TYPEFACE_WEIGHT

    (TYPEFACE_WEIGHT_INDICATOR_VALUE : in
      TYPEFACE_WEIGHT_INDICATOR_TYPE);

  --
  --more ESCAPE procedures can be inserted here
  --
end GKS_ESCAPE;
```

Date of Presentation: 9 September 1988

Sponsoring Authority: ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Set Fully Justified Text

**Description**

This escape function sets a value for the fully justified text. This value may be either on, indicating that Restricted Text (and associated Append Text) is drawn in a fully justified style, or off, indicating that Restricted Text is not drawn in a fully justified style. This attribute has no affect on the drawing of Text primitives, and does not apply to standards that do not contain a Restricted Text Primitive. See attached sheet for additional details.

**Additional Comments**

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

**Justification for Inclusion**

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable for most applications. Most proprietary graphics systems supporting typographic quality text allow fully justified text. Such text is drawn by the target system by varying the spacing between words and characters in a text string to insure that the string starts at the beginning of a restricted region, ends at the end of the region, and has a pleasing appearance. The Restricted Text primitive in the existing graphics standards partially meets this goal, but its semantics allows the target system to adjust text by many different means to meet this restriction. This escape requires that the text be fit by adjusting only the Character Spacing where this is possible. Equivalent effects cannot be achieved with typographic fonts by directly adjusting Character Expansion Factor and Character Spacing, especially where font substitution may occur. Based on "minimality", it was decided to simply add an "attribute" to Restricted Text and Append Text to affect this restriction rather than adding a separate "Justified Text" primitive.

**Relationship to Standards**

- 1) ISO 7942 (GKS) - Not applicable.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Not applicable.

Description:

**Set Fully Justified Text** sets a current fully justified text value to the value specified by the *fully justified text indicator*. This establishes whether Restricted Text (and Append Text associated with Restricted Text) are drawn in a fully justified style as defined below. This escape is applicable only to standards that include a Restricted Text primitive.

Fully Justified text is defined by these rules:

- 1) As with Restricted Text, fully justified text is constrained to be contained within the parallelogram defined with the Restricted Text primitive.
- 2) In fitting the text within this parallelogram, the values of Character Height, Character Expansion Factor, Text Precision, and TextFont Index may not be adjusted. Only the Character Spacing may be adjusted.
- 3) The text is drawn in such a way that the characters drawn appear to "fill" the parallelogram along the text path, with the first and last characters drawn touching the sides of the parallelogram. In case the text cannot be fit within the parallelogram by adjusting only the Character Spacing, then the text will be drawn using the rules for Restricted Text.
- 4) The manner in which Character Spacing is adjusted when implementing fully justified text is implementation dependent. It is suggested, however, that the excess space (the length in the Restricted Text primitive less the sum of the lengths of all the individual characters in the string) be divided between inter-character and inter-word spacing by dividing 1/3 of the excess space equally among all inter-character spaces and dividing 2/3 of the excess space equally among all inter-word spaces in the string. This is the most common rule used in typographic practice.

The following fully justified text indicator values are defined:

Off:fully justified text is set to Off, indicating that Restricted Text and associated Append Text is not drawn in a fully justified style.

On:fully justified text is set to On, indicating that Restricted Text and associated Append Text is drawn in a fully justified style.

If the fully justified text indicator value is not one of the defined values, then the default value, Off shall be used.



Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Fully Justified Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):

fully justified text indicator(Off, On) (E)

Items for Data Record:

fully justified text indicator

The following values of fully justified text indicator are defined:

1: Off

2: On

Data Record Description:

The parameter defines the fully justified text indicator.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.



**3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)**

This escape does not apply to GKS since it does not contain a Restricted Text primitive.

**4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)**

This escape does not apply to GKS since it does not contain a Restricted Text primitive.

**5) Pascal language binding (reference: ISO/IEC 8651-2 GKS Language Bindings; Part 2: Pascal)**

This escape does not apply to GKS since it does not contain a Restricted Text primitive.

**6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)**

This escape does not apply to GKS since it does not contain a Restricted Text primitive.

Date of Presentation: 18 July 1988
------------------------------------

Sponsoring Authority: ANSI
----------------------------

Class of Graphical Item: ESCAPE
---------------------------------

Specific Escape   Function Identifier:   Select Typeface Proportionate Width
--

#### Description

Select Typeface Proportionate Width selects a desired typeface proportionate width. This information is used to indicate a preference for one proportionate width over others when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. See attached sheet for additional details.

#### Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

#### Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable in office systems and graphic arts applications. Many proprietary graphics systems currently use a typographic text model that includes a proportionate width attribute. This escape allows the proportionate width of a typeface to be selected for font substitution or selection purposes. For example, extended and condensed fonts are provided, where the font designer specifies changes in character spacing to achieve the condensed or extended appearance. Although these effects could be crudely approximated by varying the Character Spacing attribute of graphical text, such an approximation is inappropriate for systems that require high-quality text.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

Description:

Select Typeface Proportionate Width sets the desired typeface proportionate width to the value specified by the typeface proportionate width indicator. This information is used to indicate a preference for one proportionate width over others when a typeface is selected during "font substitution" or when a graphical "font" is not completely specified in a typographic sense. The following proportionate width values (from ISO/DIS 9541) are defined:

- ultra condensed (highest ratio of glyph height to width)
- extra condensed
- condensed
- semi condensed
- medium
- semi expanded
- expanded
- extra expanded
- ultra expanded (lowest ratio of glyph height to width)

If the typeface proportionate width indicator is not one of the defined values, then the default value, medium, shall be used.

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Select Typeface Proportionate Width  
escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):

typeface proportionate width indicator (ultra condensed,  
extra condensed, condensed, semi condensed, medium, semi  
expanded, expanded, extra expanded, ultra expanded) (E)

Items for Data Record:

typeface proportionate width indicator

The following typeface proportionate width indicator values are  
defined:

- 1: ultra condensed
- 2: extra condensed
- 3: condensed
- 4: semi condensed
- 5: medium
- 6: semi expanded
- 7: expanded
- 8: extra expanded
- 9: ultra expanded

Data Record Description:

The parameter defines the desired typeface proportionate width.



2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

## Select Typeface Proportionate Width

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
typeface proportionate width indicator	Note 1	E

**Note 1.** Typeface proportionate width indicator is one of: (ultra condensed, extra condensed, condensed, semi condensed, medium, semi expanded, expanded, extra expanded, ultra expanded)

output data record:  
    none

#### Errors:

- 8     *GKS not in proper state: GKS shall be either in one of the states GKOP, WSOP, WSAC, or SGOP*

### 4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrS" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrS is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrS (TFPRWD)

Input Parameters:

INTEGER TFPRWD                      typeface proportionate width indicator  
                                    (ultra condensed, extra condensed,  
                                    condensed, semi condensed, medium, semi  
                                    expanded, expanded, extra expanded,  
                                    ultra expanded)

Output Parameters:  
NONE

## Select Typeface Proportionate Width

The following mnemonic FORTRAN names and their values for GKS ENUMERATION type values are added to the list in the GKS FORTRAN binding:

typeface proportionate width	ultra condensed, condensed, medium, expanded, ultra expanded	extra condensed, semi condensed, semi expanded, extra expanded,
INTEGER	GTFUC, GTFCON, GTFMED, GTFEXP, GTFUE	GTFEC, GTFSC, GTFSE, GTFEE,
PARAMETER	(GTFUC=1,  GTFCON=3, GTFMED=5, GTFEXP=7, GTFUE=9)	GTFEC=2,  GTFSC=4, GTFSE=6, GTFEE=8,

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Parameters to the Pack and Unpack Data Record functions:

INTEGER IL	1
INTEGER IA( 1 )	proportionate width indicator
INTEGER RL 0	
INTEGER SL 0	

**5) Pascal language binding** (reference: ISO DIS 8651 GKS  
Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```

GEScapeTypefaceProportionateWidthType =
    (GVEscapeUltraCondensed,
     GVEscapeExtraCondensed,
     GVEscapeCondensed,
     GVEscapeSemiCondensed,
     GVEscapeMedium,
     GVEscapeSemiExpanded,
     GVEscapeExpanded,
     GVEscapeExtraExpanded,
     GVEscapeUltraExpanded);

GREScapeDataIn = RECORD
    CASE EscapeID : GTEscapeDataTag of
        1: (
            U0001 TypeFaceProportionateWidth
            :GEScapeTypefaceProportionateWidthType);
    End;

GREScapeDataOut = RECORD
    CASE EscapeId : GTEscapeDataTag of
        1: ( ) ; (*Null Record *)
    END;

```



## Select Typeface Proportionate Width

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SELECT\_TYPEFACE\_PROPORTIONATE\_WIDTH" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for typeface proportionate width.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type TYPEFACE_PROPORTIONATE_WIDTH_INDICATOR_TYPE is (
    ULTRA_CONDENSED, EXTRA_CONDENSED, CONDENSED, SEMI_CONDENSED,
    MEDIUM, SEMI_EXPANDED, EXPANDED, EXTRA_EXPANDED, ULTRA_EXPANDED
  );
  type TYPEFACE_PROPORTIONATE_WIDTH_DATA_RECORD is
    TYPEFACE_PROPORTIONATE_WIDTH_INDICATOR_VALUE      :in
    TYPEFACE_PROPORTIONATE_WIDTH_INDICATOR_TYPE;

  procedure SELECT_TYPEFACE_PROPORTIONATE_WIDTH
    (ESCAPE_IDENTIFIER      :in ESCAPE_ID;
     TYPEFACE_PROPORTIONATE_WIDTH_INDICATOR_VALUE :in
     TYPEFACE_PROPORTIONATE_WIDTH_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

**PART 3**

**PARTIALLY REVISED PROPOSALS WHICH NEED  
ADDITIONAL TECHNICAL WORK PRIOR TO BALLOT.**



<b>Date of Presentation:</b> 10 April 1987
--

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> GDP
-------------------------------------

<b>GDP Identifier:</b> Pel Array
----------------------------------

#### Description

This Generalized Drawing Primitive provides a very flexible mechanism for describing raster (image) data within computer graphics standards. It can be viewed as an extension of the Cell Array primitive to allow alternate encoding and compression schemes. See attached sheets for additional details.

#### Additional Comments

The features in this GDP are based upon those in the Cell Array primitive, with extensions adopted from CCITT T.4 and T.6. Tag Image File Format (TIFF), and Tiled Raster Interchange Format (TRIF) as needed to meet the requirements of intended applications.

#### Justification for Inclusion

The raster data capabilities of existing graphics standards provide a rich and flexible set of facilities for dealing with raster (image) data. These facilities do, however, lack a few features that limit their wider applicability and often force users to utilize techniques outside of computer graphics standards to describe/transfer raster data. This GDP is intended to add the missing facilities. With it, a single file format—the CGM—can meet the needs of both raster and "vector" graphics storage and exchange in office, publishing, and other applications.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.
- 2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).



**Description:**

The Pel Array generalized drawing primitive is an extended version of the Cell Array primitive designed to support the transfer, storage, and printing of raster image data from a variety of sources.

**Parameters:**

3 corner points P,Q, and R (3P)

nx,ny (number of pels per line; number of lines) (2I)

encoding (one of: packed, run-length, T.4, T.6, LZW) (E)

local colour precision

pel array colour specifiers (nx\*ny\*local colour precision)

In the general case, P,Q,R can delimit an arbitrary parallelogram. P and Q delimit the end points of a diagonal of the parallelogram, and R defines a third corner.

In the simplest case, the three corner points, P,Q,R, define a rectangular area in the coordinate space. This area is subdivided into nx\*ny contiguous rectangles as follows. The edge from P to R is subdivided into nx equal intervals, and the edge from R to Q is subdivided into ny equal intervals. The grid implied consists of nx\*ny identical pels. The colour list consists of nx\*ny colour specifications, conceptually an array of dimensions nx and ny representing respectively the column and row dimensions. Array element (1,1) is mapped to the pel at corner P, and array element (nx,1) is mapped to pel at corner R. Array element (nx,ny) is mapped to the pel at corner Q. Hence, the colour elements are mapped within rows running from P to R, and with the rows incrementing in order from R to Q. [Note that all four traditional values of pel path -- 0, 90, 180, and 270 -- as well as both traditional values of line progression -- 90 and 270 -- can be realized by varying the inter-relationships among P, Q, and R]

The encoding parameter specifies how the pel array values are encoded. This parameter allows an API standard to pick up a pel array (image) obtained from an external device, such as a scanner, and image(print) it without having to interpret the data in it. Similarly, it allows a graphical metafile or interface standard to transfer such data without having to interpret it. The allowable values of encoding are:

**Packed** : No compression, but pack colour values as tightly as possible, with no unused bits except at the end of a row. The colour values are represented by rows of values, each row starting on a (16 bit) word boundary. [Note: This encoding is identical to the "packed list" mode in the binary binding of the CGM. It is also equivalent to the "packed" mode of TIFF, with byte order restricted to "MM".]

**Run-length** : The colour list values are represented by rows broken into runs of constant colour; each row starts on a (16 bit) word boundary. [Note: This encoding is identical to the "packed list" mode in the binary binding of the CGM.]

**T.4 one dimensional** : Facsimile-compatible CCITT Group 3, exactly as specified in "Standardization of Group 3 facsimile apparatus for document transmission" Recommendation T.4 Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1985, pages 16 through 21. All rows must begin on a byte boundary. The restrictions in T.4 for the number of pels per line (nx), the number of lines per pel array (ny), and the size, position, and orientation of an pel array within a picture do not apply. [Note that the addition of fill bits before EOLs to force them to end on word boundaries is not required but is allowed.]

**T.4 two dimensional** : Facsimile-compatible CCITT Group 3 two dimensional, exactly as specified in "Standardization of Group 3 facsimile apparatus for document transmission" Recommendation T.4 Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1985, pages 21 through 28. All rows must begin on a byte boundary. The restrictions in T.4 for the number of pels per line (nx), the number of lines per pel array (ny), and the size, position, and orientation of an pel array within a picture do not apply. [Note that the addition of fill bits before EOLs to force them to end on word boundaries is not required but is allowed.]

**T.6** : Facsimile-compatible CCITT Group 4, exactly as specified in "Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus", Recommendation T.6, Volume VII, Fascicle VII.3. Terminal Equipment and Protocols for Telematic Services, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1985, pages 40 through 48. The restrictions in T.4 for the number of pels per line (nx), the number of lines per pel array (ny), and the size, position, and orientation of an pel array within a picture do not apply. [Note that the addition of fill bits before EOLs to force them to end on word boundaries is not required but is allowed.] [Note: This encoding is identical to that of the T.4 two dimensional with the single exception that the "K" parameter which controls the number of consecutive two dimensional lines without an intervening one dimensional line has the value infinity rather than a small integer.]

**LZW Compression** : An adaptive compression for raster images as defined in an article by Terry A. Welch, entitled "A Technique for High Performance Data Compression", IEEE Computer, vol. 17 no. 6 (June 1984), and called the basic Lempel-Ziv & Welch (LZW) algorithm. [Note: The author's goal in that article was to describe a hardware-based compressor that could be built into disk controller or database engine, and used on all types of data. There is no specific discussion of raster images.]

LZW is fully reversible. All information is preserved, but if noise



or information is removed from an image, perhaps by smoothing or zeroing some low-order bitplanes, LZW compresses images into a smaller size. Thus, 5-bit, 6-bit, or 7-bit data masquerading as 8-bit data compresses better than true 8-bit data. Smooth images also compress better than noisy images, and simple images compress better than complex images. LZW works well on bilevel images too.

### LZW Encoding

The LZW algorithm is based on a translation table, or string table, that maps strings of input characters into codes. Variable-length codes are used, with a maximum code length of 12 bits. This string table is different for every pel array, and, remarkably, does not need to be kept for the decompressor. The trick is to make the decompressor automatically build the same table as is built when compressing the data. The following C-like pseudocode describes the coding scheme:

```
InitializeStringTable();
WriteCode(ClearCode);
 $\Omega$  = the empty string
for each character in the pel array{
    K = GetNextOctet();
    if  $\Omega$ +K is the string table{
         $\Omega$  =  $\Omega$ +K; /* string concatenation*/
    }else{
        WriteCode(CodeFromString( $\Omega$ ));
        AddTableEntry( $\Omega$ );
         $\Omega$  = K;
    }
}/*end of for loop*/
WriteCode(CodeFromString( $\Omega$ ));
WriteCode(EndofInformation);
```

The "characters" that make up LZW strings are octets of uncompressed pel array data. InitializeStringTable() initializes the string table to contain all 256 possible single octet codes, numbered 0 through 255. WriteCode() writes a code to the output stream. The first code written is a Clear code, which is defined to be code #256.  $\Omega$  represents the "prefix" string. GetNextOctet() retrieves the next octet from the input stream. The "+" sign indicates string concatenation.

AddTableEntry() adds a table entry. Since InitializeStringTable has already added 256 entries to the table, and since entry 256 is reserved for a special "clear code", and entry #257 is reserved for a special "End of Information" code, therefore the first multi-octet entry to the table is made at position 258.

Since codes are written using as few bits as possible, WriteCode() starts out with 9 bit codes since the new entries are greater than 255 but less than 512. When table entry 512 is added, WriteCode switches to 10 bit codes. Likewise, it switches to 11 bit codes at 1024 and to 12 bit codes at 2048. The table is limited to 12 bit codes, so when entry 4094 is reached, a ClearCode is written and the compresor re-initilaizes the table and starts writing out 9 bit codes again. Each encoded pel array begins with a Clear code and

ends with an End of Information code.

## LZW Decoding

The procedure for decompression is described by the following pseudocode:

```
while ((CodeNextCode() != End of Information code) {
    if ((Code == Clear code) {
        InitializeTable();
        Code = GetNextCode();
        if (Code == End of Information code )
            break;
        WriteCode(StringFromCode(Code));
        OldCode = Code;
    } /*end of Clear code case*/

    else{
        if (IsInTable(Code)) {
            WriteString(StringFromCode(Code));
            AddStringToTable(StringFromCode(OldCode)+FirstChar
                (StringFromCode(Code)));
            WriteString(OutString);
            AddStringToTable(OutString);
            OldCode = Code;

        }else{
            OutString=StringFromCode(OldCode)
                + FirstChar(StringFromCode(Code));
            AddStringToTable(OutString);
            OldCode = Code;
            WriteString(OutString);
        }
    } /*end of not-Clear code case*/
} /*end of while loop*/
```

The function `GetNextCode()` retrieves the next code from the LZW-coded data. It must keep track of bit boundaries. It knows that the first code that it gets will be 9-bit code. We add a table entry each time we get a code, so `GetNextCode()` must switch over to 10-bit as soon as string #511 is stored into the table.

The function `StringFromCode()` gets the string associated with a particular code from the string table.

The function `AddStringToTable()` adds a string to the string table. The "+" sign joining the two parts of the argument to `AddStringToTable` indicate the string concatenation.

`StringFromCode()` looks up the string associated with a given code.

`WriteString()` adds a string to the output stream.



## Pel Array

The 'local colour precision' parameter declares the precision of 'cell colour specifiers'. It applies only to computer graphics standards that support the both direct and indexed specification of colour. If indexed colour selection is used, then this parameter specifies the colour index precision. If direct colour selection is used, then then this parameter specifies the colour precision.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

The LCP (local color precision) parameter declares the precision of the cell colour specifiers. The precision is for either indexed or direct colour, according to the COLOUR SELECTION MODE of the picture. The form of the parameter is encoding dependent. If the picture uses indexed colour selection, then the form of the parameter is the same as that of COLOUR INDEX PRECISION. If the picture uses direct colour selection, then the form of the parameter is the same as that of COLOUR PRECISION. Since the array may be compressed, its length may not be able to be calculated directly from the number of pel array colour specifier values. Consequently, the pel array is treated as an array of 16 bit integer words and its length is specified by the pel array length parameter.

A functional description of the Pel Array generalized drawing primitive parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

point list(nP) contains the three points P, Q, and R  
data record(D)

Items for Data Record:

nx  
ny  
encoding  
LCP  
pel array length  
start of pel array colour specifiers  
....

Data Record Description:

The data record contains the dimensions of the pel array, its encoding type, its local colour precision, and the pel array itself.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

## Pel Array

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a and above. The corner points are transformed to NDC and the pel array through those points is then drawn. The pel array colour specifiers may be either index or direct colour values, as determined by the value of the colour specification type parameter. If the pel array uses indexed colour selection, then the local colour precision specifies the length of the index values in bits. If the pel array uses direct colour selection, then the local colour precision specifies the length of the direct RGB colour values in bits. Since the array may be compressed, its length may not be able to be calculated directly from the number of pel array colour specifier values. Consequently, the pel array is treated as an array of 16 bit integer words and its length is specified by the pel array length parameter.

A functional description of all input parameters is:

Name	Coordinate System	Values	Data Type
number of points		(3)	I
corner points (P,Q, and R)	WC		3xP
GDP identifier		as assigned	N
GDP data record: nx,ny (2I)			
encoding	(packed, run-length, T.4, T.6, LZW)		E
colour specification type	(indexed, direct)		E
local colour precision			I
pel array length			I
pel array colour specifiers (nx*ny*local colour precision)			I*pel array length

#### Errors:

- 5 GKS not in proper state: GKS shall be in the state WSAC or in the state SGOP
- 100 Number of points is invalid



## Pel Array

4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GDpqrs( N, PXA, PYA, NX, NY, IENCODE, ICSPEC, LCP,
+IPAL, IPACS) .
```

Input Parameters:

INTEGER N	number of points (3)
REAL PXA(3), PYA(3)	P, Q, and R corner points
INTEGER NX, NY	number of pels per line; number of lines
INTEGER IENCODE	encoding (packed, run-length, T.4, T.6, LZW)
INTEGER ICSPEC	colour specification type(indexed, direct)
INTEGER LCP	local colour precision
INTEGER IPEL	pel array length
INTEGER IPACS(IPEL)	pel array colour specifiers (number of values is nx*ny*local colour precision)

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

Integer IL	5 + pel array length (which depends the relationship of LCP to integer precision integer precision and encoding)
Integer IA(1)	nx
Integer IA(2)	ny
Integer IA(3)	encoding
Integer IA(4)	LCP
Integer IA(5)	pel array length
Integer IA(6)	start of pel array colour specifiers
....	
Integer RL	0
Integer SL	0

The Unpack Data Record function is not required by this escape.

5) **Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```

NumPoints = 3;
Points : GAPointArray;
GEGDPCurveProperty6 = (packed, run-length, T.4, T.6, LZW);
GEGDPCurveProperty7 = (indexed, direct);

GRGDpData = RECORD
  CASE GDPId : GTGDpDataTag OF
    1: (
      U0001 PelsPerLine           : INTEGER;
      U0001 LinesPerArray         : INTEGER;
      U0001 EncodingType          : GEGDPCurveProperty6;
      U0001 ColourSpecificationType : GEGDPCurveProperty7;
      U0001 LocalColourPrecision  : INTEGER;
      U0001 PelArrayLength        : INTEGER;
      U0001 PelArrayColourSpecifiers : array
                                     [1..PelArrayLength]
                                     of INTEGER);
  END;

```

## Pel Array

6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3: Ada)

Registered GDP's are in a library package named GKS\_GDP. GKS Ada provides a data type package, GKS\_TYPES which provides types declarations.

The binding for the "procedure PEL\_ARRAY" form (as defined in subclause 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a Pel Array.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
  type CORNER_POINTS is new WC.POINT_ARRAY (1..3);
  type ENCODING_TYPE is (PACKED, RUN_LENGTH, T.4, T.6, LZW);
  type COLOUR_SPECIFICATION_TYPE is (indexed, direct);
  type PEL_ARRAY_COLOUR_SPECIFIERS is array (NATURAL range<>) of
    ESCAPE_INTEGER);
  type PEL_ARRAY_DATA_RECORD is
    record
      PELS_PER_LINE           : ESCAPE_INTEGER;
      LINES_PER_ARRAY        : ESCAPE_INTEGER;
      ENCODING                : ENCODING_TYPE;
      COLOUR_SPECIFICATION   : COLOUR_SPECIFICATION_TYPE;
      LOCAL_COLOUR_PRECISION : ESCAPE_INTEGER;
      PEL_ARRAY_LENGTH       : ESCAPE_INTEGER;
      PEL_ARRAY              : PEL_ARRAY_COLOUR_SPECIFIERS;
    end;

  procedure PEL_ARRAY
    ( CORNER_POINTS           :in BEZIER_POINTS;
      PEL_ARRAY_RECORD       :in PEL_ARRAY_DATA_RECORD;
    );

--
--more GDP procedures can be inserted here
--

end GKS_GDP;
```

<b>Date of Presentation:</b> 9 September 1988
---

<b>Sponsoring Authority:</b> ANSI
-----------------------------------

<b>Class of Graphical Item:</b> ESCAPE
--

<b>Specific Escape Function Identifier:</b> Set Indexed Colour Response
---

#### Description

This escape function sets a value for the indexed colour response curve. This curve is used to provide exact photometric information in terms of density for indexed colour specifications contained in pel array primitives. See attached sheets for additional details.

#### Additional Comments

This escape is intended for use in conjunction with the pel array generalized drawing primitive, although it could be used for rendering other primitives as well.

#### Justification for Inclusion

Photometric information in terms of density for indexed colour specifications contained in pel array primitives is needed if output devices are to reproduce input pel arrays precisely. Many proprietary systems for defining/storing/transferring raster image data provide this capability. This is one in a set of escapes that provide extended raster/image data capabilities enabling the family of computer graphics standards to meet the requirements of office document generation/exchange and technical publications.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.



**Description:**

The purpose of the Set Indexed Colour Response escape function is to define a "curve" providing exact photometric interpretation information in terms of optical density for indexed colour image data contained in pel array primitives. In particular, if the index values represent ranges of a monochrome value such as gray, this escape can be used to provide more exact photometric interpretation information for gray scale image data. The default curve is linear in intensity/reflectance.

Since optical density is specified in terms of fractional numbers, Real numbers are used for these values. Optical densitometers typically measure densities within the range of 0.0 to 2.0. If the indexed colour response curve is known for the data in a pel array, and if the indexed colour response of the output device is known, then an intelligent conversion can be made between the input data and the output device. For example, the output can be made to look just like the input. In addition, if the input image lacks contrast (as can be seen from the response curve), then appropriate contrast enhancements can be made.

The purpose of the indexed colour response curve is to act as a "lookup" table mapping values from 0 to  $2^{*(\text{local colour precision})}-1$  into specific density values. The 0th element of the indexed colour response curve array is used to define the colour response value for all pels having an index value of 0, the 1st element of the indexed colour response curve array is used to define the colour response value for all pels having a value of 1, and so on, up to  $2^{*(\text{local colour precision})}-1$ . It is permissible to have a indexed colour response curve even for bilevel (1-bit) pel arrays. The indexed colour response curve will have 2 values.

Implementers may wish to purchase a Kodak Reflection Density Guide, catalog number 146,5947, available for \$10 or so at prepress supply houses, and use it to determine reasonable density values for their scanner or frame grabber. If this is not practical the default curve that is linear in intensity/reflectance can be used.

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM;  
Part 1: Functional Description)

The local colour precision value in this escape should match that used in subsequent Pel Array GDPs when index colour is used. A functional description of the Set Indexed Colour Response escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration  
Authority

data record (D):

LCP (local colour precision) (I)

colour response curve (  $(2^{LCP}) * R$  )

Items for Data Record:

LCP (local colour precision)

colour response curve (0)

colour response curve (1)

...

colour response curve (  $(2^{LCP}) - 1$  )

Data Record Description:

The parameters define the local colour precision and indexed colour response curve for pel array data.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

The set dash escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
LCP (local colour precision)		I
colour response curve		(2**LCP)*R
output data record:		
none		

#### Errors:

- 8     *GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP*

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrS" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrS is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrS(LCP, CRC)

Input Parameters:

INTEGER LCP	local colour precision
REAL CRC(2**LCP)	colour response curve

Output Parameters:

NONE

b) The following parameters are proposed for use when accessing this escape through the GESC function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

Integer IL	1
Integer IA(1)	LCP (local colour precision)
Integer RL	2**LCP
Real RA(1)	colour response curve (0)
Real RA(2)	colour response curve (1)
...	
Real RA(2**LCP)	colour response curve ((2**LCP)-1 )
Integer SL	0



The Unpack Data Record function is not required by this escape.

**5) Pascal language binding** (reference: ISO/IEC 8651-2 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```

GEscapeDataIn = RECORD
  CASE EscapeId : GTEscapeDataTag of
    1: (
      U0001 LocalColourPrecision: INTEGER;
      U0001 ColourResponseCurve :
        array [1..(2**LocalColourPrecision)]
          of REAL);
  END;

GEscapeDataOut = RECORD
  CASE EscapeID : GTEscapeDataTag of
    1: () ;          (*Null Record*)
  END;

```



6) **GKS Ada language binding** (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3: Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SET\_INDEXED\_COLOUR\_RESPONSE" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function to set the indexed colour response curve for a
--pel array.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type COLOUR_RESPONSE_CURVE is array
    (SMALL_NATURAL range <>) of ESCAPE_FLOAT;
  type INDEXED_COLOUR_RESPONSE_RECORD is
    record
      LOCAL_COLOUR_PRECISION :in INTEGER;
      COLOUR_RESPONSE         :in COLOUR_RESPONSE_CURVE
                               (0..(2**LOCAL_COLOUR_PRECISION-1));
    end record;

  procedure SET_INDEXED_COLOUR_RESPONSE
    (COLOUR_RESPONSE         :in INDEXED_COLOUR_RESPONSE_RECORD );
  --
  --more ESCAPE procedures can be inserted here
  --
end GKS_ESCAPE;
```

Date of Presentation: 9 September 1988
--

Sponsoring Authority: ANSI
----------------------------

Class of Graphical Item: ESCAPE
---------------------------------

Specific Escape	Function Identifier:	Set Direct Colour Response
-----------------	----------------------	----------------------------

#### Description

This escape function sets a value for the direct colour response curve. This curve is used to provide exact photometric information in terms of intensity for direct colour specifications contained in pel array primitives. See attached sheets for additional details.

#### Additional Comments

This escape is intended for use in conjunction with the pel array generalized drawing primitive, although it could be used for rendering other primitives as well.

#### Justification for Inclusion

Photometric information in terms of intensity for directly specified colour contained in pel array primitives is needed if output devices are to reproduce input pel arrays precisely. Many proprietary systems for defining/storing/transferring/viewing raster image data provide this capability. This is one in a set of escapes that provide extended raster/image data capabilities enabling the family of computer graphics standards to meet the requirements of office document generation/exchange and technical publications.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

**Description:**

The purpose of the Set Direct Colour Response escape function is to define a "curve" providing exact photometric interpretation information in terms of intensity for direct colour image data contained in pel array primitives. Three colour response curves, one each for Red, Green and Blue color information are defined. The Red entries come first, followed by the Green entries, followed by the Blue entries. The default curves are linear in intensity/reflectance. The length of each subcurve is  $2^{**}(\text{local colour precision})$ , using the same local colour precision value as subsequent pel array generalized drawing primitives. Each entry is a 16 integer value. 0 represents the minimum intensity, and 65535 represents the maximum intensity. Black is represented by (0,0,0), and white by (65535, 65535, 65535). Therefore, a color response curve entry for direct (RGB) colour data with a local colour precision of 8 bits would have  $3 * 256$  entries, each consisting of a 16 bit integer.

Relationship to particular standards:1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

The local colour precision value in this escape should match that used in subsequent Pel Array GDPs when index colour is used. A functional description of the Set Indexed Colour Response escape parameters is:

## Parameters:

function identifier (I) as assigned by the Registration Authority

## data record (D):

LCP (local colour precision) (I)  
 red response curve ( (2\*\*LCP) \* I)  
 green response curve ( (2\*\*LCP) \* I)  
 blue response curve ( (2\*\*LCP) \* I)

## Items for Data Record:

LCP (local colour precision)  
 red response curve (0)  
 red response curve (1)  
 ...  
 red response curve ((2\*\*LCP)-1 )  
 green response curve (0)  
 green response curve (1)  
 ...  
 green response curve ((2\*\*LCP)-1 )  
 blue response curve (0)  
 blue response curve (1)  
 ...  
 blue response curve ((2\*\*LCP)-1 )

## Data Record Description:

The parameters define the local colour precision and indexed colour response curve for pel array data.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.



## Set Direct Colour Response

### 3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

The set dash escape is applicable at GKS level 0a and above. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
LCP (local colour precision)		I
red response curve		(2**LCP)*I
green response curve		(2**LCP)*I
blue response curve		(2**LCP)*I
output data record:		
none		

#### Errors:

- 8     *GKS not in proper state: GKS shall be in one of the states GKOP, WSOP, WSAC, or SGOP*

### 4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in subclause 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

SUBROUTINE GEpqrs(LCP, RCRC, GCRC, BCRC)

#### Input Parameters:

INTEGER LCP                   local colour precision  
 INTEGER RCRC(2\*\*LCP)   red response curve  
 INTEGER GCRC(2\*\*LCP)   green response curve  
 INTEGER BCRC(2\*\*LCP)   blue response curve

#### Output Parameters:

NONE

## Set Direct Colour Response

b) The following parameters are proposed for use when accessing this escape through the GESC function of subclause 9.3 of the GKS FORTRAN language binding standard:

Parameters used by the Pack Data Record function for the Input Data Record:

Integer IL	$1 + 3 \cdot (2^{**}LCP)$
Integer IA(1)	LCP (local colour precision)
Integer IA(2)	red response curve (0)
Integer IA(3)	red response curve (1)
...	
Integer IA( $2^{**}LCP+1$ )	red response curve ( $(2^{**}LCP)-1$ )
Integer IA( $2^{**}LCP+2$ )	green response curve (0)
Integer IA( $2^{**}LCP+3$ )	green response curve (1)
...	
Integer IA( $2 \cdot 2^{**}LCP+1$ )	green response curve ( $(2^{**}LCP)-1$ )
Integer IA( $2 \cdot 2^{**}LCP+2$ )	blue response curve (0)
Integer IA( $2 \cdot 2^{**}LCP+3$ )	blue response curve (1)
...	
Integer IA( $3 \cdot 2^{**}LCP+1$ )	blue response curve ( $(2^{**}LCP)-1$ )
Integer RL	0
Integer SL	0

The Unpack Data Record function is not required by this escape.

### 5) Pascal language binding (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in subclause 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GREscapeDataIn = RECORD
  CASE EscapeId : GTEscapeDataTag of
    1: (
      U0001 LocalColourPrecision : INTEGER;
      U0001 RedResponseCurve
        : array [1..(2**LocalColourPrecision)] of INTEGER;
      U0001 GreenResponseCurve
        : array [1..(2**LocalColourPrecision)] of INTEGER;
      U0001 BlueResponseCurve
        : array [1..(2**LocalColourPrecision)] of INTEGER;
    )
  END;

GREscapeDataOut = RECORD
  CASE EscapeID : GTEscapeDataTag of
    1: ( ) ;
  END;
(*Null Record*)
END;
```

## Set Direct Colour Response

6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3: Ada)

Registered ESCAPE's are in a library package named GKS\_ESCAPE. GKS Ada provides a data type package, GKS\_TYPES which provides type declarations.

The binding for the "procedure SET\_DIRECT\_COLOUR\_RESPONSE" form (as defined in subclause 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function to set the indexed colour response curve for a
--pel array.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type DIRECT_COLOUR_RESPONSE_CURVE is array
    (SMALL_NATURAL range <>) of INTEGER;
  type DIRECT_COLOUR_RESPONSE_RECORD is
    record
      LOCAL_COLOUR_PRECISION :in INTEGER;
      COLOUR_RESPONSE         :in DIRECT_COLOUR_RESPONSE_CURVE
        (0..(2**LOCAL_COLOUR_PRECISION-1));
    end record;

  procedure SET_DIRECT_COLOUR_RESPONSE
    (COLOUR_RESPONSE         :in DIRECT_COLOUR_RESPONSE_RECORD );
  --
  --more ESCAPE procedures can be inserted here
  --
end GKS_ESCAPE;
```

PART 4

NEW PROPOSALS SUBMITTED FOR THE FIRST TIME THIS YEAR.





**Proposal Number:**

**Date of Presentation:** 6 October 1989

**Sponsoring Authority:** ANSI

**Class of Graphical Item:** ESCAPE

**Specific Escape Function Identifier:** Segment List

**Description**

This escape function defines an association between the name of a global segment (in a CGM) and an identifier. This identifier is used by the receiving system to locate the segment. Its purpose is to allow such segments to be externally defined and included in a CGM file by reference. This escape applies only to the CGM standard.

**Additional Comments**

None.

**Justification for Inclusion**

In many "closed" interchange environments it is useful to build libraries of standard graphical objects that can be incorporated into pictures by reference. This leads to more compact picture descriptions and to more uniform appearance.

**Relationship to Standards**

- 1) ISO 7942 (GKS) - Does not apply.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Does not apply.

Description:

**Segment List** permits global segments to externally defined. This escape is only applicable to the CGM standard. Each segment name in the *segment name list* is associated with the corresponding identifier in the *external identifier list*. The total number of elements in both lists must be identical and is given by the *number of correspondences*. The effect is as if each segment definition that is externally referenced were included within the metafile descriptor as a global segment definition.

The content and structure of the identifiers in the external identifier list are not standardized by this escape element, and are expected to be defined by application profiles. However, the following scheme is suggested for general use in CGMs that do not conform to a specific application profile:

- 1) each identifier is a string consisting of three portions;
- 2) the portions are separated from each other by the solidus character (/);
- 3) the first portion is the local file name of a CGM file where the segment may be found;
- 4) the second portion gives the picture number within the file designated in the first portion; a value of zero (0) designates a global segment within that file;
- 5) the third portion gives either the local segment name within the picture designated by the second portion or the global segment name in case the second portion designates a global segment.

For example, "My Symbol File/0/1" designates global segment "1" in file "My Symbol File". (Note that global segment names are realized as integers in the CGM, so character codes in identifiers are translated into equivalent integer values as necessary to determine their meanings.)

If any segment name or identifier is invalid, then that element of both lists shall not be processed and no association shall be established. If the number of correspondences is invalid, the escape shall be ignored. If the number of correspondences and the contents of the two lists are not consistent, then the escape shall be ignored. If any identifier cannot be translated into a valid external reference, then that element of both lists shall be ignored.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Select General Fill escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):

number of correspondences (the number of elements in both  
the segment name list and the external identifier list)  
segment name list (a list of segment names)  
external identifier list (a list of external identifiers)

Items for Data Record:

number of correspondences (I)  
segment name list (nSN)  
external identifier list (nS)

Data Record Description:

The parameters designate a correspondence between a set of segment names and a set of external identifiers.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7492, GKS Functional Specification)

The proposer only wishes to use this escape with the CGM standard.

4) GKS FORTRAN language binding (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

The proposer only wishes to use this escape with the CGM standard.



Segment List

5) GKS Pascal language binding (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The proposer only wishes to use this escape with the CGM standard.

6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

The proposer only wishes to use this escape with the CGM standard.

Proposal Number:

Date of Presentation: 6 October 1989

Sponsoring Authority: ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Glyph Association

#### Description

This escape function defines an association between a glyph collection and a set of codes. Such an association is commonly called a "character set." This escape is applicable only to the CGM standard. The designated tail sequence is used to invoke defined "character set" within a CGM. The glyph collection can be either a registered glyph collection. Glyph identifiers may be registered identifiers or may have only private meaning.

#### Additional Comments

None.

#### Justification for Inclusion

Presently, the CGM standard allows only coded character sets defined or registered according to the procedures of ISO 646, ISO 2022, or ISO 2375 to be used in text strings. In office systems standards the use of such coded character sets is being replaced by a new set of glyph identifier and glyph collection identifier registration procedures that de-couple the notion of glyph (or "character") from the code used to represent that glyph in interchange. Without this extension, these glyphs and glyph collections cannot be used within a CGM, since the associated registers do not record any association of glyphs to codes. Further, there are additional requirements for user defined fonts and user defined character sets that can conveniently utilize an escape that allows an association between codes and glyphs to be defined within a CGM. This escape is designed to meet both these needs.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Does not apply.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Does not apply.
- 4) ISO 10036 (Procedure for registration of glyph and glyph collection identifiers) - Identifiers registered according to these procedures can be used in this Escape function.

Description:

**Glyph Association** permits the definition of "character sets" constructed by associating glyph identifiers with points in a code table (in the sense of ISO 2022). This escape is only applicable to the CGM standard. The number of bytes in each code word is defined by *code bytes*. Each code in the *code list* is associated with the corresponding glyph identifier in the *glyph identifier list*. The total number of elements in both lists must be identical and is given by the *number of glyphs*. The *designation tail sequence* must correspond to one of the character sets listed in the CHARACTER SET LIST element, and establishes the relationship between the character set type in that element and the codes in this escape function. The *glyph collection identifier* may be used to associate the glyph collection with its registered identifier for information purposes only. The *default glyph identifier* designates the glyph that is used for all unspecified bit combinations (codes).

The content and structure of the identifiers in the *glyph identifier list* are not standardized by this escape element, and may be defined by application profiles or taken from those in the International Register of Glyph Identifiers established by ISO 10036. The *glyph collection identifier* is either a registered glyph collection name or is a null string.

The number of bytes in a code point in both this element and in the CHARACTER SET LIST must be consistent. If they are not, this element shall be ignored. If the number of correspondences and the contents of the two lists are not consistent, then this escape shall be ignored. If any glyph identifier is not recognized, then the corresponding code shall be set the default glyph identified by the *default glyph identifier*. The *code list* need not contain a code point for each possible bit combination. Those code points that are omitted shall use the default glyph identified by the *default glyph identifier*. If the *default glyph identifier* is not recognizable, the asterisk glyph (\*) shall be used.



Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM;  
Part 1: Functional Description)

A functional description of the Select General Fill escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):

number of glyphs (the number of elements in both code list and glyph identifier list)

code bytes (the number of bytes in each element of code list)

code list (a list of codes)

glyph identifier list (a list of glyph identifiers)

designation tail sequence (the tail sequence used to designate this character set in the CGM)

glyph collection identifier (the registered identifier - if any - of the collection of glyphs in the glyph identifier list)

default glyph identifier (the identifier of a default glyph that is used for unspecified code points)

Items for Data Record:

number of glyphs	(I)
code bytes	(I)
code list	list of (S)
glyph identifier list	list of (S)
designation tail sequence	(S)
glyph collection identifier	S
default glyph identifier	S

Data Record Description:

The parameters define a glyph collection and associate each glyph in the collection with a code.



**2) CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

**3) GKS Functional Specification** (reference ISO 7492, GKS Functional Specification)

The proposer only wishes to use this escape with the CGM standard.

**4) GKS FORTRAN language binding** (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

The proposer only wishes to use this escape with the CGM standard.

**5) GKS Pascal language binding** (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The proposer only wishes to use this escape with the CGM standard.

**6) GKS Ada language binding** (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

The proposer only wishes to use this escape with the CGM standard.

Proposal Number:

Date of Presentation: 6 October 1989

Sponsoring Authority: ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Select General Fill

#### Description

This escape function selects a general fill interior style for filled area primitives. It is only applicable to the CGM standard. The filled area treatment in this case is derived by copying the indicated segment, repeated as necessary, throughout the filled area. The current Fill Reference Point and Pattern Size are used to locate the segment and modify its size. The interior style reverts back to one of the defined interior styles when the next Interior Style element is encountered.

#### Additional Comments

None.

#### Justification for Inclusion

The exchange of technical illustrations and engineering drawings both require filled area treatments that cannot be implemented with one of the present styles. In particular, the ability to register hatch styles seems to be limited to hatches consisting of parallel lines only. Patterns cannot be implemented efficiently on many devices and are inadequate for filling areas with geometrically-described pictures. What is required is a filled area interior style treatment that: (i) allows any "picture" as a fill; and (ii) the filled area does not transform as the object is re-scaled. The use of such a general fill can replace most purposes for which clipping to arbitrary boundaries might be used, while still allowing devices that cannot support such clipping to produce a reasonable picture.

#### Relationship to Standards

- 1) ISO 7942 (GKS) - Does not apply.
- 2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.
- 3) ISO 8651 (GKS Language Bindings) - Does not apply.

Description:

Select General Fill selects interior style "general fill" based on the segment identified by segment identifier as defining the general fill. This escape is only applicable to the CGM standard. This segment is copied within subsequent filled areas as many times as needed to fill them completely. This copying takes place according to the following rules:

1) The graphical effect is as if successive COPY SEGMENT Elements were encountered with:

a) the initial translation portion of the copy transformation matrix translates to the current Fill Reference point;

b) subsequent translations are derived by calculating the extent of the virtual device coordinates present in the segment, and offsetting the translation point to "repeat" the segment as necessary until the area is completely filled;

c) the scaling and rotation portion of the copy transformation matrix is derived from the current Pattern Size; no rotation is done, and the pattern height and pattern width components define the vertical and horizontal scaling respectively;

2) All graphical objects are clipped to the boundary of the filled area primitive.

If the segment identifier is not a valid segment name, then to current interior style shall remain unchanged. The interior style reverts to one of the interior styles defined in the CGM standard when the next valid INTERIOR STYLE element is encountered.



Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Select General Fill escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):

segment indicator (the name of a segment used as the fill)

Items for Data Record:

segment indicator (SN)

Data Record Description:

The parameter gives the name of the segment to be used as a "general fill".

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) **GKS Functional Specification** (reference ISO 7492, GKS Functional Specification)

The proposer only wishes to use this escape with the CGM standard.

4) **GKS FORTRAN language binding** (reference ISO/IEC 8651-1, GKS Language Bindings; Part 1: FORTRAN)

The proposer only wishes to use this escape with the CGM standard.



5) GKS Pascal language binding (reference: ISO/IEC 8651-2, GKS Language Bindings; Part 2: Pascal)

The proposer only wishes to use this escape with the CGM standard.

6) GKS Ada language binding (reference ISO/IEC 8651-3, GKS Language Bindings; Part 3:Ada)

The proposer only wishes to use this escape with the CGM standard.

NIST-114A  
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE  
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER

NISTIR 4330

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE

May 1990

4. TITLE AND SUBTITLE

Graphics Standards in the Computer-Aided Acquisition and Logistic Support (CALS) Program, Fiscal Year 1989, Volume 2: MIL-D-28003 Revisions, CGM Registration

5. AUTHOR(S)

Daniel R. Benigni, Editor

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE  
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY  
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

Progress Report 10/88-9/89

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

Office of the Secretary of Defense  
Production and Logistics/Systems/CALS  
Room 3B322, Pentagon  
Washington, D.C. 20301-8000

10. SUPPLEMENTARY NOTES

☐ DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The Computer-aided Acquisition and Logistic Support (CALS) Program is a DoD Industry strategy to transition from paper-intensive acquisition and logistic processes to a highly automated and integrated mode of operation for the weapon systems of the 1990s. These volumes document the accomplishments of the Graphics Software Group of the National Institute of Standards and Technology (NIST) in support of computer graphics standards for CALS in FY89. They provide a progress report on continuing graphics standards efforts related to the Computer Graphics Metafile (CGM standard). These reports are divided into two volumes:

1, Test Requirements Document and Extended CGM (CGEM); and 2, MIL-D-28003 Revisions and CGM Registration.

Volume 2. Efforts to update and revise MIL-D-28003 are detailed. The work accomplished to meet CALS requirements for functionality beyond that of the current CGM, through the standards process of graphical registration, is also documented.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

application profile; CALS; CGEM; CGM; CGM registration; conformance testing; DoD; graphics; illustration data; MIL-D-28003; test requirements document

13. AVAILABILITY

☒

UNLIMITED

☐

FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

☐

ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE,  
WASHINGTON, DC 20402.

☒

ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES

329

15. PRICE

A15







